Departamento de Ingeniería de Sistemas y Automática

PROYECTO FIN DE CARRERA

REGISTRO AUTOMÁTICO DE NUBES DE PUNTOS

Autor: Francisco Jesús Pérez Pérez

Director: Javier Victorio Gómez González

Tutor: Santiago Garrido Bullón

Leganés, Diciembre de 2016

Título: Registro automático de nubes de puntos
Autor: Francisco Jesús Pérez Pérez
Director: Javier Victorio Gómez González

EL TRIBUNAL

Presidente: Higinio Rubio Alonso

Vocal: David Álvarez Sánchez

Secretario: Jorge Muñoz Yañez-Barnuevo

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 19 de Diciembre de 2016 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

Quisiera agradecer a todas aquellas personas que me han apoyado para lograr terminar mis estudios, en especial a mis padres, que siempre han confiado en mi y han soportado la mayor carga, también al resto de mi familia, y todas aquellas personas que de corazón me han ayudado a conseguirlo, con especial mención a Marta por su gran apoyo y dedicación.

Índice

Índice de figuras

1 Introducción

El ser humano ha tenido, a lo largo de la historia, el afán de crecer, lo que ha fomentado el desarrollo industrial y con ello el bienestar. Todo ello ha repercutido en la sociedad, de forma que las tareas realizadas por el ser humano han sido sustituidas por máquinas. En la Primera Revolución Industrial, con la máquina de vapor, se pudieron reemplazar las tareas de empleo de fuerza bruta y las de los movimientos repetitivos, esto influyó en la velocidad de los procesos consiguiendo mayor producción. Además, se incorpora el transporte, tanto marítimo como terrestre, a mayor escala. Aparecen nuevos empleos, en los cuales el factor humano tiene un valor añadido, con jornadas laborales con menor carga de trabajo y mayor productividad.

La electricidad cambió la forma de fabricar, la sociedad y la economía. Las nuevas comodidades hicieron que la demografía explotase, surgiendo grandes empresas capaces de dirigir el desarrollo y la prosperidad. La forma de vida fue cambiando y haciendo surgir una nueva conciencia social.

La industria siempre está en constante cambio debido a los procesos que se realizan, nunca son definitivos sino que están inacabados. Las máquinas fueron capaces de sustituir una primera fase al factor humano, aunque solo eran capaces de realizar tareas repetitivas, sin poder tomar decisiones. Actualmente, en algunos casos, toman decisiones para procesar la materia prima.

Este hecho, es lo que ha fomentado el desarrollo de los denominados robots, utilizándolos en entornos industriales, como por ejemplo la industria del automóvil. Los robots no solo son capaces de operar estos espacios, han sido desarrollados para que trabajen de forma eficaz en entornos abiertos e interactuando con seres humanos.

En las últimas décadas ha existido el deseo de implementar robots que sean capaces de desarrollar tareas en entornos domésticos y en la vida diaria de los usuarios. Aunque la tecnología de la que inicialmente se disponía era demasiado costosa y poco estable. Los robots que se han desarrollado se han limitado a hacer sencillas tareas en entornos de laboratorio donde se disponía de costosos sensores.

El principal problema en los robots es reconocer el medio sobre el que tienen que realizar los desplazamientos y sus tareas. Inicialmente había que recurrir a sensores de capturas de imagen que solo estaban disponibles en 2D y posteriormente en 3D, pero en ambos casos es importante obtener una imagen en 3D. Para realizar tareas complejas y conocer el entorno espacial se necesitan algoritmos con altos requerimientos de procesamiento.

En los últimos años se han desarrollado sensores de captura de imagen en 3D de bajo coste, accesibles a un público con más bajo poder adquisitivo. Fomentando que se realicen diferentes plataformas en código abierto para el desarrollo de estas tecnologías.

2 Objetivos

El principal objetivo es construir una imagen 3D de un espacio interior completo, utilizando una cámara de tipo estereoscópica como la Kinect. Para ello, se realizará un sistema de visión artificial basado en tecnología de nubes de puntos con software libre y gratuito. Este debe cumplir los requisitos para entregar la información suficiente que permita reconocer el entorno. Para reconocer el medio es necesario obtener varias capturas de imágenes para agruparlas bajo un mismo sistema de referencia. Se estudiarán los procesos de solapamiento, buscando el más adecuado en función del tiempo requerido para calcularlo.

Los sensores de capturas de imágenes en 3D son capaces de generar gran cantidad de puntos, lo que supone el almacenamiento de gran cantidad de datos que en algún momento deben ser procesados. Para resolver el problema se tomarán tácticas que puedan reducir los datos con el mismo nivel de información. La información generada debe tener una estructura sencilla para que los sistemas que necesiten acceder a ella sean capaces de procesarla con agilidad.

3 Estado de arte

3.1 Visión artificial

La realización de tareas, en el ser humano cómo en los animales, necesitan examinar el entorno, con el fin de tomar decisiones y estrategias que le ayuden a conseguir los objetivos propuestos. Algunos toman diferentes estrategias de exploración, como por ejemplo: los búhos son capaces de emitir un sonido y logran moverse durante la noche. La gran mayoría de los animales disponen del aparato visual, que se estima que ocupa el 70% de las tareas del cerebro humano en el análisis de la información visual. La visión humana es el sentido más desarrollado y el que menos se conoce debido a su gran complejidad. Es una actividad inconsciente y difícil de saber cómo se produce. Hoy en día, se carece de una teoría que explique cómo los humanos perciben el exterior a través de la vista. La visión artificial es una rama de la inteligencia artificial que tiene por objetivo modelar matemáticamente y entender el entorno, que le rodea a una máquina, capaz de realizar tareas de forma autónoma.

Para el estudio de la visión artificial se necesitan medios de captura de imágenes. Pueden ser obtenidas: en escala de grises como en el inicio de la fotografía o en colores distinguiéndose diferentes espacios como RGB o CMYK. Las imágenes pueden ser con diferentes estructuras de datos más allá de los formatos:

1. La forma más simple es en dos dimensiones, este método intenta replicar el sistema de percepción visual del ojo humano.

2. Imágenes en tres dimensiones, donde además de las características de color se obtiene la característica de la profundidad de cada dato, inicialmente, para resolver el problema de obtención de imágenes en 3D, se planteó replicar el sistema de percepción visual del ojo humano, mediante la toma sucesiva de imágenes desde varios puntos, o con dos cámaras en lugares diferentes, en la actualidad existen métodos que controlan patrones o parámetros de luz para calcular la distancia.

Las tareas de percepción de un sistema de visión artificial, no solo tienen que medir y detectar el medio sino reconocer e interpretar el medio que los rodea. En un entorno industrial, donde existe la posibilidad de lograr mejoras, la visón artificial ha hecho que se hayan involucrado áreas como la informática, la óptica, la ingeniería mecánica y la automatización industrial, trabajando con sistemas de captura de imagen, dispositivos de entrada/salida y redes de ordenador para el control de equipos destinados a la fabricación. Se destinan para tareas en las cuales el ser humano no es capaz de tener una constancia suficiente para garantizar la calidad del trabajo, como es la inspección visual de alta velocidad, gran aumento, funcionamiento las 24 horas o repetibilidad de medidas. En un enfoque académico, existe la necesidad de llegar a realizar multitud de tareas que transformen el medio o simplemente desplazarnos a través del mundo, en ambos casos de la forma más óptima

posible.

El desarrollo de la inteligencia artificial se originó desde 1960, donde se planteaba el problema de cómo una máquina era capaz de interactuar con el medio. Desde entonces surgieron diferentes ramas, entre ellas la visión artificial, donde a lo largo de la historia los investigadores más reputados la han definido como:

1. Ballard&Brown (1982): la Visión Artificial es la construcción de descripciones de objetos físicos, llenas de significado y explícitas, a partir de imágenes. Incluyen el procesamiento de imágenes (no se considera que sea acertado en la actualidad).

2. Horn (1986) : la entrada a un sistema de visión por máquina es una imagen o varias imágenes mientras que la salida es una descripción que debe cumplir dos criterios: estar relacionada con la imagen observada y contener toda la información que se necesita para realizar una tarea determinada. El procesamiento de imágenes es una parte de esta tarea.

3. Low (1991): el procesamiento de imágenes es el procesado de una imagen por una computadora, con el objetivo de producir otra imagen, mientras que la visión artificial trata de la adquisición, el procesado, la clasificación, el reconocimiento y en su conjunto la toma de decisiones posterior al reconocimiento.

4. Nalwa (1993): la Visión Artificial describe la deducción automática de las estructuras y propiedades de un mundo tridimensional, posiblemente dinámico, a partir de una o varias imágenes bidimensionales de él. El autor excluye de su libro el reconocimiento de objetos basado en modelos, ya que, según él, este reconocimiento es un proceso cognitivo y no de percepción.

5. Trucco&Verri (1998): la búsqueda de propiedades del mundo 3D a partir de una o más imágenes digitales. Es importante notar que en este texto sí se incluye el reconocimiento de objetos.

6. Hoffman (1998): tal vez estemos en una época en la que el reconocimiento de objetos sea parte de la visión artificial, pero no sería extraño que en el futuro asistiéramos a una separación de esta materia de la visión artificial como observamos el procesamiento y análisis de imágenes. El mundo de la visión artificial es fundamentalmente un mundo 3D.

4 Tecnologías asociadas

La capacidad para desarrollar un proyecto tecnológico en visión artificial, hace necesaria unas herramientas que son costosas, si bien como trabajamos con licencias de software abierto y libre tenemos la oportunidad de obtener el software con coste cero. El hardware necesario para el proyecto, tenemos la oportunidad de utilizar diferentes tecnologías, debido a las implementaciones realizadas en el software libre.

4.1 Software

4.1.1 Lenguaje C++

C++ evolucionó de C, que a su vez evolucionó de dos lenguajes de programación anteriores, BCPL y B. En 1967, Martin Richards desarrolló BCPL como un lenguaje para escribir software de sistemas operativos y compiladores para sistemas operativos. Ken Thompson modeló muchas características en su lenguaje B, a partir del trabajo de sus contrapartes en BCPL, y utilizó a B para crear las primeras versiones del sistema operativo UNIX, en los laboratorios Bell en 1970.

El lenguaje C evolucionó a partir de B, gracias al trabajo de Dennis Ritchie en los laboratorios Bell. C utiliza muchos conceptos importantes de BCPL y B. Inicialmente, se hizo muy popular como lenguaje de desarrollo para el sistema operativo UNIX. En la actualidad, la mayor parte de los sistemas operativos se escriben en C y/o C++. C está disponible para la mayoría de las computadoras y es independiente del hardware. Con un diseño cuidadoso, es posible escribir programas en C, que sean portables para la mayoría de las computadoras.

Por desgracia, el amplio uso de C con diversos tipos de computadoras (a las que a veces se les denomina plataformas de hardware) produjo muchas variaciones. Este fue un grave problema para los desarrolladores de programas, quienes necesitaban escribir programas portables que pudieran ejecutarse en varias plataformas. Era necesaria una versión estándar de C. El Instituto Nacional Estadounidense de Estándares (ANSI) cooperó con la Organización Internacional para la Estandarización (ISO) para estandarizar C a nivel mundial; el documento estándar colectivo se publicó en 1990 y se conoce como ANSI/ISO 9899: 1990.

C99 es de los estándar ANSI más reciente para el lenguaje de programación C. Se desarrolló para evolucionar el lenguaje C, que pudiera estar al corriente con el poderoso hardware de la actualidad y con los requerimientos cada vez más exigentes de los usuarios. El estándar C99 es más capaz de competir con los lenguajes como Fortran. Las capacidades de C99 incluyen el tipo long long para las máquinas de 64 bits, los números complejos para las aplicaciones de ingeniería y un mayor soporte para la aritmética de punto flotante. C99 también hace a C más consistente con C++, al permitir el polimorfismo a través de funciones matemáticas de tipo genérico, y por medio de la creación de un tipo booleano definido.

A principios de la década de los ochenta, Bjarne Stroustrup desarrolló una extensión de C en los laboratorios Bell: C++. Este lenguaje proporciona un conjunto de características que pulen al lenguaje C, pero lo más importante es que proporciona la capacidad de una programación orientada a objetos.

Una revolución se está gestando en la comunidad del software. Escribir software rápida, correcta y económicamente es aún una meta escurridiza, en una época en que la demanda de nuevo y más poderoso software se encuentra a la alza. Los objetos son en esencia, componentes reutilizables de software, que modelan elementos del mundo real. Los desarrolladores de software están descubriendo que el uso de una metodología de diseño e implementación modular, orientada a objetos puede hacerlos más productivos, que mediante las populares técnicas de programación anteriores. Los programas orientados a objetos son más fáciles de entender, corregir y modificar.

4.1.2 Herramientas de desarrollo

Los lenguajes de programación necesitan ser compilados para poder ser ejecutados en una máquina con su correspondiente sistema operativo. Cuando trabajamos con lenguaje C++ se requieren de una colección de compiladores GCC o la herramienta de gestión de proyectos de software multiplataforma Cmake.

4.1.2.1 Colección de compiladores GNU

Figura 4.1: logotipo de GCC

La colección de compiladores GNU[1] (GNU Compiler Collection),son unas herramientas libres creadas por el proyecto GNU de forma libre y liberadas bajo la licencia GPL. Estos compiladores son básicos en los sistemas basados en UNIX, tanto libres como propietarios (como es el caso del Mac OS X de Apple).

GCC nació como parte del proyecto GNU, creando un compilador exclusivamente de C. Con el tiempo se fue extendiendo a más lenguajes, entre los que se encuentra C++ o Python. El proyecto GNU se inició en los años ochenta con Richard Stallman como máximo responsable, con el fin de crear un sistema operativo totalmente libre.

Actualmente GCC soporta los lenguajes más comunes, como C, C++, Java o Fortran, mientras

1 El logotipo de GCC es cortesía de gcc.gnu.org

que de forma no estándar se añaden otros como Mercury o VHDL. También soporta multitud de plataformas diferentes además de la x86 clásica de los ordenadores. Por ejemplo, tiene soporte para ARM (tipo de arquitectura muy extendida entre dispositivos móviles como teléfonos o tabletas) o PowerPC (arquitectura que actualmente solo se encuentra en videoconsolas o estaciones para cálculos complejos). Además es el compilador integrado en multitud de famosos entornos de desarrollo multiplataforma, como Eclipse, Netbeans o Code::Blocks.

4.1.2.2 Cmake

CMake[2] fue creado en respuesta a la necesidad de un potente entorno de compilación multiplataforma, para Herramientas de Registro, Segmentación y Percepción (ITK). Financiado por la Biblioteca Nacional de Medicina para proyecto de análisis de imagen. Fue influenciado por un sistema anterior, llamado PCMAKER, creado por Ken Martin y otros desarrolladores para apoyar las Herramienta de visualización (VTK) de código abierto para sistemas de visualización y gráficos 3D. Para crear CMake, Bill Hoffman en Kitware incorporó algunas ideas clave de PCMAKER, y ha añadido muchas ideas propias, para adoptar algunas de las funcionalidades de la herramienta de configuración de Unix.

Figura 4.2: logotipo de CMake

La implementación inicial CMake fue a mediados de 2000, acelerando su desarrollo principios de 2001. Muchas de esas mejoras fueron influenciadas por otros desarrolladores que incorporan CMake en sus propios sistemas. Por ejemplo: la comunidad de software *Vision-something-Libraries*(VXL) adoptó CMake, como su entorno de compilación, contribuyendo muchas características esenciales.

Brad Rey añade varias características con el fin de apoyar a CABLE, GCC-XML, GE Corporate R&D requiere apoyo de su infraestructura de pruebas (DART). Se añadieron otras funciones de apoyo a la transición del entorno de compilación de VTK para CMake y apoyar ParaView, un sistema de visualización paralela apoyado por el Advanced Computing Lab en Los Alamos National Laboratory.

Para poder usar Cmake correctamente es necesario crear un fichero de nombre CMakeLists.txt, que contendrá los parámetros necesarios, como ficheros que forman parte de la compilación, librerías que hay que incluir o el nombre final de la aplicación. CMake lee dicho archivo y genera un proyecto adecuado al Sistema Operativo, compilador elegido y la ubicación de las librerías que van a ser enlazadas. Como extra es posible generar un Makefile específico para algunos entornos de desarrollo concretos, como Visual Studio o Eclipse.

Entre otras características, permite analizar dependencias para los lenguajes C, C++, Fortran y Java. También permite el uso de compilaciones paralelas o cruzadas, además genera ficheros

2 Logotipo de Cmake es cortesía de cmake.org

Makefile específicos para algunos entornos de desarrollo.

4.1.3 Bibliotecas

En términos de programación, una biblioteca o librería es el conjunto de implementaciones funcionales que se emplean para desarrollar otros programas, proporcionando servicios que pueden facilitar el desarrollo. Las implementaciones se enlazan con las librerías mediante llamadas en el código fuente, lo que hace que a la hora de compilar dichas bibliotecas se unan en el ejecutable. Son utilizadas constantemente en programación.

Todos los lenguajes poseen sus propias librerías, que facilitan tareas tan habituales y comunes como la muestra de elementos por pantalla. Una de las grandes facetas de C++ es la de poder utilizar conjuntos de librerías externas, pero programadas en él, creadas por terceros, proporcionando un abanico de posibilidades, que en este proyecto serán aprovechadas para simplificar el trabajo.

La mayoría de los sistemas operativos modernos proporcionan bibliotecas que implementan los servicios del sistema. De esta manera, estos servicios se han convertido en una *materia prima* que cualquier aplicación moderna espera que el sistema operativo ofrezca. La mayor parte del código utilizado por las aplicaciones modernas se ofrece en estas bibliotecas.

Dentro de las librerías, nos encontramos con dos tipos fundamentales: estáticas y dinámicas. En sus orígenes, las librerías solo podían ser estáticas, estando integradas en el propio archivo ejecutable. Este paso se realizaba al compilar, ya que el compilador en lugar de crear un enlace simbólico, las incluía dentro del ejecutable. Concretamente, convertía todas las direcciones no resueltas en fijas, cargando tanto el código como las bibliotecas en posiciones de memoria, en el tiempo de ejecución. Dicho proceso de enlazado es lento y es necesario cada vez que algún módulo es modificado.

Por otro lado y como avance reseñable en este campo, nos encontramos con las librerías dinámicas. En este caso se cargan en tiempo de ejecución, en lugar de hacerlo en tiempo de compilación. A la hora de compilar, el proceso de enlazado es muy rápido, solo crea enlaces simbólicos a la posición de la librería, en lugar de enlaces estáticos.

Las bibliotecas dinámicas proporcionan un mayor rendimiento en los programas, unas mejoras en el tiempo de compilación y una reducción en el tamaño de los ejecutables, suelen ser la opción preferente a la hora de programar.

4.1.3.1 Point Cloud Library

Figura 4.4: logotipo de Point Cloud Library(PCL)

PCL[3] es una moderna biblioteca de C ++ con plantilla completas para el procesamiento de nubes de puntos 3D. Escrito teniendo en cuenta el rendimiento y la eficiencia de las CPU modernas, las estructuras de datos subyacentes de PCL, hacen uso de fuertes optimizaciones en SSE. La mayoría de las operaciones matemáticas se implementan con Eigen, una biblioteca de plantillas de código abierto para álgebra lineal .

Además, PCL proporciona soporte para OpenMP (ver http://openmp.org) e Intel Threading Building Blocks (TBB) biblioteca para la paralelización de procesamientos múltiples. La columna vertebral de las operaciones de búsqueda rápida de vecinos más cercanos (k-nearest) es proporcionada por FLANN (Biblioteca Fast Library for Approximate Nearest Neighbors). Todos los módulos y algoritmos en PCL utilizan punteros compartidos usando Boost, evitando así la necesidad de volver a copiar los datos que ya está presente en el sistema.

Desde el punto de vista algorítmico, PCL tiene la intención de incorporar una multitud de algoritmos de procesamiento de 3D que operan con los datos de nubes de puntos, entre ellos: el filtrado, la estimación de función, reconstrucción de la superficie, ajuste del modelo, segmentación, registro, etc.

Cada conjunto de algoritmos se define a través de las clases base, tratan de integrar toda la funcionalidad común que se utiliza en toda la tubería, manteniendo así las implementaciones de los algoritmos reales compactas y limpias. La interfaz básica para la ejecución de dicho tratamiento en PCL es:

1. Crear un objeto de procesamiento (filtro, segmentación, keypoint).

2. Utilizar 'setInputCloud' para pasar la nube de puntos de entrada al módulo de procesamiento.

3. Establecer algunos parámetros.

4. Llamar a la función de cálculo para obtener la salida.

Para simplificar aún más el desarrollo, la PCL se divide en una serie de bibliotecas de código más pequeñas, que pueden ser compiladas por separado:

1. Filtros: implementa filtros de datos, tales como la disminución de resolución, extracción de

3Logotipo de Point Cloud Library cortesía de pointclouds.org

índices, proyecciones, etc.

2. Features: implementa muchas características 3D como superficies normales y curvaturas, punto de estimación de frontera, momentos invariantes, curvaturas principales, descriptores PFH y FPFH, espín de imágenes, imágenes integrales, descriptores NARF, RIFT, RSD, VFH, SIFT en los datos de intensidad.

3. IO: implementa operaciones I/O tales como escritura o lectura de archivos PCD(Point Cloud Data).

4. Segmentation: implementa la extracción de clúster, ajuste del modelo a través de métodos de consenso de muestra para una variedad de modelos paramétricos (planos, cilindros, esferas, líneas, etc.), la extracción de prisma poligonal, etc .

5. Surface: implementa técnicas de reconstrucción de superficies, mallado, envolvente convexa, Moving Least Squares, ect.

6. Registration: implementa métodos de registro de nubes de puntos tales como ICP, ect.

7. Keypoints: implementa diferentes métodos de extracción de puntos clave, que pueden ser usados paso de preprocesamiento para decidir donde aplicar descriptores de características.

8. KdTree: Es un conjunto de librerías que usando a su vez la librería FLANN(librería de C++ especializada en búsquedas de puntos próximos en espacios de grandes dimensiones) obtiene como resultado unas búsquedas eficientes y rápidas de elementos cercanos. Un Kd-tree, o árbol de k dimensiones, es una estructura de datos espacial que engloba un conjunto de puntos de kdimensiones permitiendo una búsqueda eficiente de sus puntos próximos.

9. Octree: Esta librería proporciona métodos para generar estructuras de datos jerárquicas. Esto habilita una forma de partición espacial, simplificación y búsqueda de operaciones en los conjuntos de datos de puntos. Cada uno de los Octree tiene ocho nodos, que no necesariamente son derivados de él. El nodo principal describe un cubo delimitador que encapsula todos los puntos. Por cada nivel del árbol este espacio se subdivide en el factor de 2 que resulta tras incrementar la resolución del voxel. La correcta implementación de Octree facilita el trabajo con puntos próximos y provee multitud de algoritmos implementados que realizan este tipo de tareas.

10. Visualization: Son librerías diseñadas para la visualización de los datos tridimensionales de la nube de puntos. Las rutinas que utiliza dibujan imágenes bidimensionales de forma que den la sensación de tener tres dimensiones. La librería incluye funciones de renderizado y configuración de propiedades, tipo colores, tamaño de los puntos, etc. También trae métodos para dibujar una estructura tridimensional básica en pantalla, visualización de histogramas y

finalmente, visualización de range images. Se incluye una aplicación, llamada pcd_viewer, que representa por pantalla nubes de puntos guardadas en el formato PCD, pudiendo modificar parámetros en tiempo real como la representación del color o el tamaño.

11. Common: Contiene las estructuras de datos y métodos comunes, utilizados por la mayor parte de las librerías PCL. Entre otras cosas, incluye las estructuras de las posiciones, los colores o la representación de puntos. Además de funciones para calcular distancias, medias o transformadas geométricas.

Para garantizar la exactitud de las operaciones en PCL, los métodos y las clases en cada una de las librerías mencionadas, contiene test unitarios y de regresión. El conjunto de pruebas unitarias se compilan por demanda y verificado, con frecuencia, por una comunidad dedicada al desarrollo. Los respectivos autores de un componente específico, son informados inmediatamente cuando un componente falla un test. Esto asegura que cualquier cambio en el código se prueba a fondo y cualquier nueva funcionalidad o modificación no lo romperá, depende de PCL.

4.1.3.2 OpenCV

Figura 4.5: logotipo de OpenCV

OpenCV[4] es una biblioteca libre de visión artificial originalmente desarrollada por Intel. Desde que apareció su primera versión alfa en el mes de enero de 1999, se ha utilizado en infinidad de aplicaciones. Desde sistemas de seguridad con detección de movimiento, hasta aplicaciones de control de procesos donde se requiere reconocimiento de objetos. Esto se debe a que su publicación se da bajo licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación con las condiciones en ella expresadas.

OpenCV es multiplataforma, existiendo versiones para GNU/Linux, Mac OS X y Windows. Contiene más de 500 funciones que abarcan una gran gama de áreas en el proceso de visión, como reconocimiento de objetos (reconocimiento facial), calibración de cámaras, visión estéreo y visión robótica.

El proyecto pretende proporcionar un entorno de desarrollo fácil de utilizar y altamente eficiente. Esto se ha logrado realizando su programación en código C y C++ optimizados, aprovechando además las capacidades que proveen los procesadores multinúcleo. OpenCV puede además utilizar el sistema de primitivas de rendimiento integradas de Intel, un conjunto de rutinas de bajo nivel específicas para procesadores Intel.

4 Logotipo de OpenCV cortesía de opencv.org

4.1.3.3 Boost

Figura 4.6: logotipo de librería Boost

Boost[5] está compuesta por un conjunto de bibliotecas, de software libre y revisión de pares, diseñadas para aumentar las capacidades del lenguaje de programación C++. Si bien, el lenguaje utilizado tiene una librería estándar, en la cual están analizadas y comprobadas para que su funcionamientos sea estable. Sin embargo, en Boost el desarrollador que trabaje con esta librería puede tener mayores posibilidades, al tener muchas más herramientas que en la biblioteca estándar. En el proyecto de esta librería se trabaja bajo una licencia tipo BSD, lo que permite ser utilizada en cualquier tipo de proyecto, ya sean comerciales o no.

4.2 Adquisición de datos

Por definición, nos referiremos al conjunto de puntos 3D, como *nube de puntos* con estructura P. Representa el formato de entrada de datos básicos para un sistema de percepción 3D, de entrada discreta, con representaciones significativas del mundo circundante. Sin ninguna pérdida de generalidad. Las coordenadas $\{x_i,y_i,z_i\}$del punto $p_i \in P$, con respecto a un sistema de coordenada, el origen está situado en la posición desde la cual se tomó la imagen. Cada punto mide la distancia en los tres ejes de coordenadas desde el punto de vista del sensor a la superficie.

Para la toma de adquisición de nube de puntos o datos 3D, tenemos diferentes tecnologías que se basan en:

1. Time-of-flight
2. Modelo Pin-hole
3. Visión estereoscópica
4. Sistemas estereoscópico de codificación luz

En el anexo I se expondrá los métodos de adquisición de nubes de puntos, así como los dispositivos comerciales que los utilizan. Esto nos ayudará a entender las dificultades a las que nos enfrentamos, según el sistema que se utilice.

Para el desarrollo del proyecto se utilizará un sensor asequible en el mercado como es la Kinect, si bien como se desarrolla en el anexo I este es adecuado para entornos interiores donde la

5 Logotipo de librería Boost cortesía de boost.org

luz solar no interfiere con la emitida por el sensor. Además la precisión del sensor para distancias muy superiores a 7 metros es ineficaz, debido al elevado error de los datos obtenidos.

4.3 Formato de datos

La información recogida por los sensores de capturas de imágenes 3D, envían la información para ser guardada o procesada desde una unidad de computación. Esta información puede ser dispuesta con múltiples formatos. Los gráficos por ordenador y las comunidades de desarrollo de geometría computacional en particular, han creado numerosos formatos para describir polígonos arbitrarios y nubes de puntos adquiridos utilizando escáneres láser. Estos son algunos de los formatos más estándar:

1. PLY : formato de archivo basado en polígonos, desarrollado en la Universidad de Stanford por Turk et al.
2. STL : formato de archivo nativo para el software CAD estereolitografía creado por 3D Systems.
3. OBJ: formato de archivo de definición de geometría primero desarrollado por Wavefront Tecnologías.
4. X3D : el formato de archivo ISO estándar basado en XML para representar datos de gráficos 3D.

4.3.1 Formato PCD

Los formatos estándar están orientados para la utilización de diferente tecnología de gráficos 3D. Realmente se cumplen los requisitos que exigía las tecnologías para las que fueron diseñados, donde el formato resultaba eficaz. Sin embargo, para el caso de la percepción artificial surge la necesidad de crear un formato con el que puedan elaborar nuevas técnicas, las cuales realizan procesos muy diferentes a las tecnologías a las que se asocia el formato de imagen 3D.

Para poder realizar las tareas de procesado en 3D, se trabajará con conjunto de puntos P, distribuidos en el espacio cada punto p_i, contará al menos con las coordenadas cartesianas en el espacio euclídeo (x_i,y_i,z_i). Además, en este formato se podrá recoger la información de color en cada punto, tanto en formato RGB como en escala de grises o intensidad. También se puede incluir información que se ha procesado después de la captura, como los vectores normales a cada punto y otras características. Otro de los puntos fuertes de este formato es la cabecera, donde se impone como se distribuyen los datos, teniendo mayor flexibilidad para tener diferentes tipos de datos en cada nube. Se pueden recoger los datos de forma matricial indicando las filas y las columnas, esto puede facilitar algoritmos heredados de las imágenes 2D. En la siguiente tabla se expone un ejemplo de cómo es la cabecera de un archivo .pcd donde se almacena una nube de puntos.

1	# .PCD v0.7 - Point Cloud Data file format
2	VERSION 0.7
3	FIELDS x y z rgba
4	SIZE 4 4 4 4
5	TYPE F F F U
6	COUNT 1 1 1 1
7	WIDTH 640
8	HEIGHT 480
9	VIEWPOINT 0 0 0 1 0 0 0
10	POINTS 307200
11	DATA ascii
12	-0.72405237 -0.61462384 1.9150001 4470583
13	-0.72416663 -0.61783338 1.9250001 4012599
14	-0.71683335 -0.61783338 1.9250001 4076856
15	-0.72049999 -0.61783338 1.9250001 3488055
.	.
:	:

5 Fundamentos teóricos

En el siguiente capítulo se explicarán los fundamentos teóricos que se deben tener en cuenta para enfocar el proyecto, que en su mayoría son necesarios para el procesamiento de nubes de puntos. La mayor parte de los fundamentos teóricos son ideas heredadas del procesamiento de imágenes digitales o visión artificial en 2D, las tecnologías de captura de nubes de puntos son recientes y por lo tanto, el desarrollo de algoritmos de procesamiento. Todos los estudios realizados que se tengan en cuentas se obtendrán mediante las funciones de la biblioteca Point Cloud Library y sus tecnologías compatibles.

5.1 Características

Las nubes de puntos que captaron los sensores dan la información de distribución de las superficies del entorno. Los puntos por si solos no aportan información, por lo que es necesario estudiar las posiciones de sus vecinos, para extraer la información de ellos. Los tipos de características se clasifican en función de los niveles de dependencia con otros puntos.

1. Características intrínsecas: son las debidas al punto, y no tienen relación con los puntos circundantes si bien, pueden estar afectados por el sensor de captura, son únicos para cada punto, son las coordenadas del punto así como los niveles de intensidad de gris o de color.

2. Características locales: este tipo de datos es obtenido a través de una máscara o vecindad de puntos cercanos los cuales influyen en su valor, existen unidades vectoriales como la normal, escalares como la curvatura o en forma de matriz como la matriz de covarianza.

3. Características globales: ayuda a formar un conjunto de puntos de la nube que tiene una forma geométrica dada, como puede ser un plano, una esfera o una recta.

Los puntos son obtenidos por medio de un sensor, que captura la profundidad de ellos y en algunos casos el color, pero las características también se refiere a los parámetros que pueden aportar los próximos a ellos. Este tipo de características serán explicadas en las siguiente secciones.

5.1.1 Vector normal.

Es conocido que los sensores capturan nubes de puntos representan la superficie que existe en el entorno, delimitando el mundo sólido. Las superficies se pueden estudiar a nivel diferencial en el punto, obteniendo diferente información. La información puede ser trasladada a un punto de la nube, matemáticamente tienen el mismo significado.

La mayoría de los métodos intenta estimar un vector normal para cada punto de la nube. Este es un procedimiento razonable, especialmente si el vector normal resultante se integra en un espacio de características en el que cada elemento de datos corresponde a un punto de la nube. Consideramos un conjunto de n puntos $P = \{p_1, p_2, ..., p_n\}$, $p_i \in \mathbf{R}^3$, que estará representado por una

matriz de datos

$$P=\left[p_1,\ \cdots,\ p_n\right]^T\ , \tag{5.1}$$

donde $\mathbf{p}_i$= $[p_{ix}, p_{iy}, p_{iz}]^T$ representa las coordenadas de 3D de los puntos medidos, para todo punto $\mathbf{p}_i$ se estimará su vector normal $\mathbf{n}_i=[n_{ix}, n_{iy}, n_{iz}]^T$, desde un conjunto de puntos vecinos $\mathbf{Q}_i=\{\mathbf{q}_{i1},\mathbf{q}_{i2}, \ldots ,\mathbf{q}_{ik}\}$, donde $\mathbf{q}_{ij} \in P$, $\mathbf{q}_{ij} \neq \mathbf{p}_i$ en adelante nos referiremos $\mathbf{Q}_i=[\mathbf{q}_{i1},\mathbf{q}_{i2}, \ldots ,\mathbf{q}_{ik}]^T$ y

$\mathbf{Q}_i^+=[\mathbf{p}_i,\mathbf{q}_{i1},\mathbf{q}_{i2}, \ldots ,\mathbf{q}_{ik}]^T$ donde $\mathbf{Q}_i$ es la matriz de vecinos y $\mathbf{Q}_i^+$ es la matriz de vecinos ampliada con todos los vecinos además del propio punto $\mathbf{p}_i$.

Para calcular el vector normal existen diferentes métodos, estos requieren obtener previamente los puntos vecinos $\mathbf{Q}_i$. Esta tarea necesita comprobar todos los puntos de la nube, verificando las distancias entre $\mathbf{q}_{ij}$ y $\mathbf{p}_i$, si esta es menor al umbral impuesto se aceptará el punto como parte del subconjunto, para el cálculo del vector normal. El método utilizado en PCL para el cálculo de la normal, hace uso del método clásico de estimación de un plano $S_i = n_{ix}\, x + n_{iy}\, y + n_{iz}\, z + d$ para un punto en $\mathbf{Q}_i^+$ lo que sería resolver:

$$\min_{b_i} \left\| \left[Q_i^+ \ 1_{1+k} \right] b_i \right\|_2 \tag{5.2}$$

donde 1_m representa un vector de $m \times 1$ de unos $y = [n^T_i, d]^T\ b_i$. Este problema de optimización se puede solucionar mediante el cálculo por descomposición de valores singulares SVD ($U\Sigma V^T$) de $\mathbf{Q}^+_i$. El b_i minimizador es el vector en V que corresponde al valor singular más pequeño en Σ(matriz con los valores singulares) y por tanto el vector normal n_i se obtiene directamente.

5.1.2 Geometrías

Cuando se captura una imagen en 3D se mide la distancia que hay del sensor de rango a una superficie, donde este punto representa una pequeña superficie, tratada como un punto. El conjunto de puntos obtenidos forman una envolvente sobre la proyección desde el punto de vista. En esta envolvente se contienen múltiples formas geométricas, algunas más sencillas y otras más complejas. La idea es conseguir crear conjuntos de puntos que forman parte de una de las figuras, para evitar tratar la imagen a través de estos conjuntos de forma local. Si el conjunto de puntos es tratado por los parámetros de una figura geométrica mantenemos la misma información con un conjunto de datos inferior.

La búsqueda de figuras geométricas es una tarea compleja en función del número de puntos que contenga la nube. Este proceso es realizado a través de los algoritmo como RANSAC. Esta herramienta nos permitirá formar subconjuntos de puntos de la nube, ya sea para segmentar planos o en otros casos líneas. La búsqueda de diferentes formas geométricas permite reducir el volumen de datos de la nube.

5.2 Filtrado de nubes de puntos

La adquisición de imágenes de profundidad tiene ciertos problemas según la tecnología que se utilice en el procedimiento. En el procesamiento de imágenes digitales en 2D, aparecen datos erróneos, un problema que ha sido heredado en la tecnología para adquisición de nubes de puntos en 3D, por lo que alguno de sus algoritmos también han sido heredados. Por lo tanto, al existir una serie de problemas análogos al procesamiento de imágenes en 2D, se han propuesto mecanismos teóricos similares para la solución de los mismos.

El filtrado es muy importante para el procesamiento digital. Forma parte de la primera etapa, según los resultados obtenidos o los cambios aplicados se obtienen una serie de resultados que parecen satisfactorios. EL filtrado supone corregir aquellos puntos que han sido perturbados realizando operaciones de transformación que lo sitúen en una posición más verosímil en función de los puntos cercanos o suprimiendo el elemento.

Para un detalle más profundo de los tipos de filtro que se pueden utilizar en las nubes de puntos se recomienda ir al anexo II donde se detalla la base matemática necesaria para realizar el proceso. Por otro lado, en el anexo III se detallan las librerías de PCL donde se definen algunos filtros disponibles, como se comportan para diferentes umbrales y los tiempos de cómputo. Del anexo III se deduce que los tiempos de cómputo son cortos, pues los umbrales necesario para una reducción fuerte no son muy grandes. Si bien es obvio, que entre mayor es el umbral mayores serán los tiempos de procesado. Cabe destacar que las clases desarrolladas no siempre tienen el comportamiento que se espera, donde se crean vecindades con diferentes densidades de puntos.

5.3 Keypoint

Los puntos de interés es una terminología reciente en la visión artificial, que se refiere a los puntos obtenidos por una secuencia de procesamiento. Un punto de interés es un punto de la imagen que en general tiene las siguientes características:

1. Tienen una clara definición, preferiblemente con base matemática, bien fundada.
2. Tiene una posición bien definida en el espacio.
3. La estructura local de la imagen alrededor del punto de interés es rica en términos de información local, de manera que el uso de los puntos de interés simplifica su posterior procesamiento en el sistema de visión.
4. Es estable bajo perturbaciones locales y globales en el dominio de la imagen como las variaciones de iluminación o brillo, tal que los puntos de interés puedan ser calculados de

forma fiable con alto grado de reproducción.

Históricamente, la noción de puntos de interés se remonta a la detección de esquinas en imágenes 2D, donde las características estaban en los primeros trabajos de detección. Con el objetivo principal de obtener un detector robusto, estable y bien definido. Para el seguimiento y reconocimiento de objetos en tres dimensiones desde imágenes en dos dimensiones. En la práctica, la mayoría de los detectores de esquinas son sensibles, pero no específicamente a las esquinas, sino a las regiones locales de la imagen, que tienen un alto grado de variaciones en todas las direcciones. El uso de puntos de interés también se remonta a la noción de las regiones de singulares, que han sido utilizados para señalar la presencia de objetos.

En los nuevos trabajos de desarrollo de visión artificial en 3D los puntos de interés se están denominando “puntos clave” o “keypoint”, obtenidos con métodos heredados 2D. Algunos algoritmos de detectores están implementados específicamente para utilizarlos en nubes de puntos 3D. Hay que diferenciar entre detectores que se implementan con principios de imágenes 2D, y los que tienen los originados específicamente para nubes de puntos 3D. Además se pueden utilizar detectores para imágenes 2D sobre nubes que deben estar ordenadas.

Los keypoint debe cumplir las siguientes condiciones:

1. Escasez: por lo general un pequeño conjunto de puntos son keypoint

2. Repetibilidad: si un punto ha sido determinado como keypoint en un conjunto de puntos, debe ser en otra nube de puntos similar (estos puntos son llamados estables).

3. Distinción: el área que rodea cada punto debe tener una forma o apariencia única que se refleja en los descriptores.

Si se precisa de conocimientos sobre los diferentes algoritmos de extracción de keypoint en el anexo II se explica los fundamentos teóricos necesarios para hallar puntos de interés. En este anexo se clasifican los diferentes tipos que existen y se expone la base matemática necesaria para realizar el cálculo, así como otros aspectos necesarios para su comprensión. Por otro lado, en el anexo III se comparan los diferentes métodos de extracción, comparando parámetros como la repetibilidad y los tiempos de cómputo.

5.4 Descriptores

En su representación nativa, los puntos se definen en el concepto de sistemas de cartografía 3D simplemente están representados mediante sus coordenadas cartesianas x, y, z, con respecto a un origen determinado. Suponiendo que el origen del sistema de coordenadas no cambia con el tiempo,

podría haber dos puntos p_1 y p_2, adquiridos en los instantes t_1 y t_2, que tiene las mismas coordenadas. Comparar estos puntos es un problema mal planteado, porque a pesar de que son iguales con respecto a alguna medida de distancia (por ejemplo, métrica euclídea), podrían ser muestreados en diferentes superficies, y por lo tanto, representan diferente información, cuando se toman junto con otros puntos circundantes de sus proximidades. Es debido, a que no hay garantías de que el mundo no ha cambiado entre t_1 y t_2. Algunos dispositivos de adquisición pueden proporcionar información adicional, para un punto de vista, tal como un valor de intensidad, textura,y/o un color. Sin embargo, no resuelve el problema por completo y la comparación sigue siendo ambigua.

Cuando se necesita comparar puntos, por diversas razones, se requiere conocer características y métricas que sean capaces de distinguir entre las superficies geométricas. El concepto de un punto 3D como una entidad singular con coordenadas cartesianas desaparece. Un nuevo concepto de descriptor local toma su lugar. La literatura es abundante de diferentes esquemas de asignación de nombres que describen la misma conceptualización, como descriptores de forma o características geométricas, pero nos referiremos como representaciones de entidades punto.

Mediante la inclusión de los vecinos de más cercanos, la geometría de la superficie muestreada subyacente puede ser inferida y capturada en la formulación descriptores, lo que contribuye a la solución del problema de ambigüedad en comparaciones. Idealmente, los descriptores resultantes serían muy similares (con respecto a alguna métrica) para los puntos que residen en los mismos o en superficies, y diferente para puntos que se encuentran en diferentes superficies. Una buena representación de entidad de punto se distingue de una mala, por ser capaz de capturar las mismas características de la superficie de la zona en la presencia de:

1. Transformaciones rígidas: las rotaciones en 3D y traslaciones 3D en los datos no deben influir en la resultante vector de características F estimación.

2. Densidad variable de muestreo: es considerado como un parche local muestreado más o menos densamente deben tener el mismo vector firma de características.

3. Ruido: la representación de entidad de puntos deberá conservar las mismas o muy similares valores en su vector de características en la presencia de ruido suave en los datos.

Los descriptores representan la geometría con la que se distribuyen los puntos con respecto al punto de consulta. Existen dos tipos de descriptores según la influencia de los puntos que estudien:

1. Descriptores locales: se caracterizan por tomar como referencia el punto de consulta, estudiando los puntos vecinos más cercanos. La información que se obtiene es la que se confina cerca del punto, dentro de una esfera que abarque los suficientes puntos para generar información.

2. Descriptores globales: los puntos de consulta son comparados con referencias de la imagen,

estas son obtenidos a través de todos los puntos. Por tanto, la información es relativa a la geometría del conjunto de la nube de puntos.

Los descriptores suelen aportar la información de un pequeño conjunto de puntos, porque no todos los puntos tienen información relevante. Solo se estudian aquellos puntos de los que suponemos especial interés por su singularidad, son los denominados keypoints. Estos pueden ser calculados mediante tres parámetros, con lo que los podemos clasificar como:

1. Relación de distancias: se estudia cómo se distribuyen los puntos, teniendo en cuenta las distancias con un o varios puntos. En algunos casos se estudian los triángulos que se forman en un mallado previo.

2. Relación entre normales: lo más habitual es estudiar la relación entre vectores, ya sea entre dos normales de diferentes puntos, o también estos con el vector que los une.

3. Relación por colores: este método solo es admitido para descriptores locales. Pues estudia los cambios de color que se producen en la máscara. Este tipo de descriptor es heredado de visión 2D.

5.5 Correspondencias

La correspondencia consiste en emparejar los puntos de las muestras de dos conjuntos de nubes que pretenden ser superpuestas, para ello se analizan y comparan características de los puntos de esta. Dependiendo del tipo de característica, podemos utilizar diferentes métodos para encontrar la correspondencia.

Para puntos coincidentes (usando las coordenadas xyz de los puntos como características) existen diferentes métodos para datos organizados y no organizados.

1. Correspondencias mediante fuerza bruta.

2. Búsqueda mediante el método kd-tree al vecino más cercano.

3. Búsqueda en el espacio de la imagen de datos organizados.

4. Búsqueda en el espacio de índices de datos organizados.

Para características coincidente (no usar las coordenadas de los puntos, sino ciertas características) solo existen los siguientes métodos:

1. Fuerza bruta de coincidencias.

2. Búsqueda mediante el método kd-tree al vecino más cercano,

Además para la búsqueda tenemos dos tipos de estimación de correspondencia se distinguen:

1. La estimación de la correspondencia directa (por defecto) busca correspondencias en la nube B para cada punto en la nube A.

2. Estimación de correspondencia "*Recíproca*" busca las correspondencias de la nube A con la nube B y de B a A, y solo usa las intersecciones.

Para un conocimiento más extenso sobre las correspondencias, puede dirigirse al anexo II, donde se explica los fundamentos teóricos del funcionamiento de las correspondencias.

5.6 Filtrado de correspondencias

Las correspondencias no válidas pueden afectar negativamente el resultado de registro, la mayoría de los procesos de registro cuentan con un paso de rechazo. Se trata de filtrar los pares de puntos emparejados en la etapa anterior con el fin de facilitar el algoritmo de estimación de la transformación hacia la convergencia al mínimo global. Este paso tiene la ventaja de disponer de información auxiliar de las nubes de puntos de entrada, como normales locales o estadísticas sobre correspondencias. Alguno de los métodos de rechazo de correspondencia son:

1. *Rechazo de correspondencia basada en la distancia*: este método filtra los pares de puntos con una distancia mayor que un umbral dado . Se utiliza en la formulación original del algoritmo ICP.

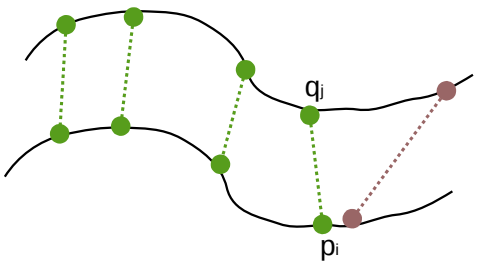


Figura 5.1: rechazo basado en la distancia.

2. El rechazo basado en distancia media: a diferencia del de rechazo anterior, no utiliza un umbral fijo. Calcula como la media de todas las distancias de punto a punto en el conjunto de entrada de las correspondencias. Considera la distribución de las distancias entre los puntos y se adapta a ella, cada vez es más pequeña como las dos nubes de puntos se acercan

durante las iteraciones ICP. En comparación con un umbral adaptativo el basado en el valor medio, la mediana es a menudo más eficaz en la reducción de la influencia de los valores atípicos.

3. Rechazo de pares con destinos duplicados: cada punto muestreado en la nube fuente se asigna a una correspondencia en la nube de destino. Puede ocurrir que un punto en la nube de destino se asigna múltiples puntos de origen correspondiente. Este supresor solo mantiene un ejemplo par ($p_{i_{min}}$, q_j), el que tiene la distancia mínima entre todos los pares $\{(p_i, q_j)\}$

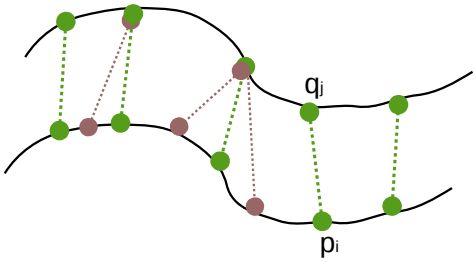


Figura 5.2: rechazo basado en pares repetidos.

4. El rechazo basado en la compatibilidad de la normal: este filtro utiliza la información de los puntos relativa a sus normales y rechaza aquellos pares que tienen las normales inconsistentes, es decir, el ángulo entre las normales es mayor que un umbral dado. Se puede rechazar pares erróneos, que parecen correctos cuando se juzga solo por la distancia entre los puntos.

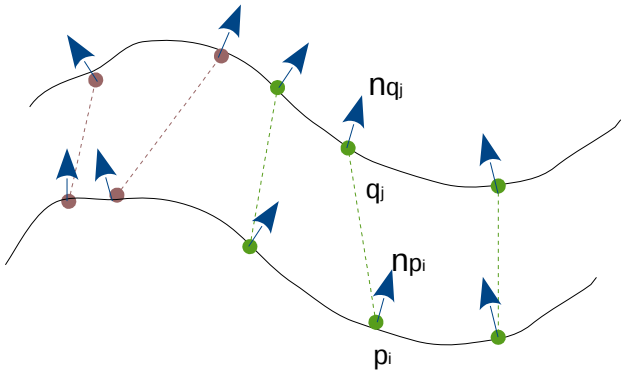


Figura 5.3: rechazo basado en compatibilidad entre normales.

5. El rechazo basado en las fronteras de la superficie: las nubes de puntos capturadas por un sensor de proyección, que representan superficies que tienen superposición parcial, si se acepta correspondencias que contienen puntos de los límites de la superficie puede introducir errores. Con el fin de detectar tales puntos, podemos explotar la naturaleza organizada de los mapas de profundidad y eliminar las correspondencias que contienen puntos sobre las discontinuidades de profundidad, moviendo una ventana a través de la imagen de profundidad y de comprobar si hay suficientes puntos en la ventana que están dentro de un umbral profundidad desde el punto central.

6. Rechazo basado en RANSAC: este método aplica Random sample consensus (RANSAC) para estimar una transformación de subconjuntos del conjunto dado y elimina las correspondencias de valores atípicos en base a la distancia euclídea entre los puntos, después de la transformación calculada, se aplica a la nube de puntos de origen. Es muy eficaz para mantener el algoritmo ICP convergiendo en mínimos locales(tiene sentido si tenemos en cuenta el elevado numero de puntos suprimidos, que puede generar optimos diferentes al del registro real), ya que siempre se producen ligeras diferencias emtre correspondencias y es bueno el filtrado de valores atípicos. Además, proporciona buenos parámetros iniciales, para el cálculo de la transformación con todas las correspondencias. El registro de nubes de puntos procedentes de sensores proyectivos, existen rechazo de correspondencia que aprovechan la estructura de imágenes como de los datos de entrada, esta sería una variante especial 2D de la correspondencia de rechazo basado en RANSAC, que descarta pares en función de su distancia de píxeles en el plano de la imagen, después de que el punto de origen se transforma y se proyecta en el plano de la cámara de destino.

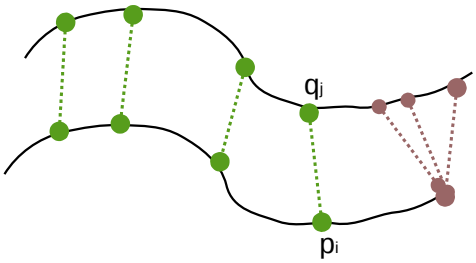


Figura 5.4: rechazo basado en las fronteras de la superficie.

7. Multiprocesos de rechazo: muy a menudo, no se aplica solo un único supresor, sino varios que rechazan correspondencia, se ponen en cola a fin de aplicar un proceso de filtrado, por ejemplo: una combinación de diferentes soluciones para rechazar pares de puntos , haciendo hincapié en la importancia de rechazar puntos en los límites de malla. Se eliminan pares en base a su compatibilidad normal y en función de su distancia de punto a punto utilizando un umbral de distancia basado en la distancia mediana.

La utilización de filtro de correspondencias no es obligatorio para conseguir el registro de dos nubes de puntos, aunque es recomendable imponer este filtro en los algoritmos, no se recomienda imponer umbrales muy estrictos, pues podría suceder que obtengamos un conjunto de pares vacíos.

5.7 Estimación de transformación

La toma de imágenes se realiza desde diferentes puntos de vista, cuando los datos son organizados en un formato la referencia del origen están en el punto de captura. El objetivo final del

registro es organizar las nubes con un único origen en común, diferente del punto de captura de cada una de las imágenes.

El paso final del registro, es la transformación de los puntos para tener un origen común, para esto es necesario obtener una matriz de transformación que sea capaz de tener en cuenta el movimiento de traslación y rotación que se puede producir en la cámara de forma rígida.

A través de los años, se han propuesto diferentes enfoques matemáticos para resolver la matriz de transformación rígida T que minimiza el error de los pares de puntos. T se compone de una rotación R y una traslación T. Teniendo en cuenta que cuando se hace referencia a una transformación T y un punto p, se utilizarán las coordenadas homogéneas.

Hay dos métricas de error principales a ser minimizadas que se han considerado en la literatura: punto a punto y de punto a plano, donde (p_k, q_k) es el k-ésimo de los N pares de correspondencias de la nube de fuente a la nube de destino.

$$E_{punto\,a\,punto}(T)=\sum_{k=1}^{N} w_k \|\boldsymbol{T}\, p_k - q_k\| \tag{5.3}$$

$$E_{punto\,a\,plano}(T)=\sum_{k=1}^{N} w_k \left((\boldsymbol{T}\, p_k - q_k)\cdot n_{q_k}\right)^2 \tag{5.4}$$

El parámetro w_k es opcional, puede utilizarse para la ponderación de los pares con el fin de concederles más o menos importancia en la formulación de mínimos cuadrados (w_k = 1 sino se aplica ponderación):

1. Error de punto a plano: Chen y Medioni [3] introdujo el error de punto a plano (Ec. 5.4) y demostró que es más estable, converge y más rápido que los métodos anteriores. Se utiliza la distancia entre el punto fuente p_k, el plano descrito por el punto de destino q_k y de su superficie local normal n_{q_k}. A diferencia del error de punto a punto, no tiene una solución de forma cerrada, por lo que la minimización se realiza mediante soluciones no lineales (tal como Levenberg-Marquadt, según lo propuesto en [4]), o por linealización [5] (bajo el supuesto de ángulos de rotación pequeños, es decir, seno(θ) ~ θ y cos ~ 1).

   Dependiendo de la superficie subyacente y la distribución de los puntos, utilizando el error de punto a plano puede ser considerablemente más robusto. Un procedimiento estándar para minimizar que se basa en el optimizador no lineal de Levenberg Marquardt [4].

2. Error estándar de punto a punto: el error estándar utilizado en el algoritmo ICP, es el error de punto a punto. Fue mencionado por primera vez por Arun [1]; investigadores propusieron diversas formas de minimizar el mismo, seguido de la introducción del algoritmo ICP Eggert et al.[2] evaluando cada uno de estos métodos en términos de estabilidad y precisión numérica para llegar a la conclusión de que son similares.

3. Error ponderado de punto a plano: asignar un peso diferente a cada correspondencia puede mejorar la convergencia. La ponderación de los pares de puntos puede ser visto como un rechazo suave. El ajuste de la influencia de la correspondencia de puntos ruidosos en el proceso de minimización. La ponderación puede ser: en función de la distancia de punto a punto, punto a plano entre los puntos, en función del ángulo entre las normales correspondientes a los puntos o una función del modelo de ruido del sensor que ha sido utilizado. Numerosas publicaciones [6, 7, 8, 9] consideran la ponderación de los pares de puntos, como un paso beneficioso para la tasa de robustez y la convergencia del algoritmo de estimación de transformación.

4. Ponderación lineal de mínimos cuadrados de punto a plano: de manera similar a la anterior, se aplica mínimos cuadrados de punto a plano variante, donde se puede utilizar los pesos no constantes.

5. Error generalizado y Generalizado-ICP: Segal et al. [10] propuso un error que generalizado punto a punto y punto a plano, que utiliza las covarianzas de los puntos más cercanos Q ($\sum_i^P$ y $\sum_i^Q$) con el fin de alinear las superficies subyacentes en lugar de los puntos :

$$E_{Generalizado-ICP}(\boldsymbol{T}) = \sum_{k=1}^{N} \boldsymbol{d}_k^{(\boldsymbol{T})^T} \left(\sum_k^Q + \boldsymbol{T} \sum_k^P \boldsymbol{T}^T \right)^{-1} \boldsymbol{d}_k^{(\boldsymbol{T})} \quad (5.6)$$

donde $\boldsymbol{d}_k^{(\boldsymbol{T})} = \boldsymbol{q}_k - \boldsymbol{T}\, p_k$ son las distancias de punto a punto de las correspondencias (p_k, q_k), T es la matriz de transformación y T^T su transpuesta.

Para los algoritmos de registro iterativos, los criterios de terminación del proceso de refinamiento para la transformación deben ser definidos. Muchas veces simplemente se fija un número máximo de iteraciones, pero existen muchos otros criterios que se pueden especificar, como:

1. Umbral de transformación relativa: específica la diferencia en la transformación máxima de una iteración a la siguiente, que se considera lo suficientemente pequeña para que el optimizador converja.

2. Número máximo de iteraciones similares: los criterios de parada anteriores tiene la desventaja que un minimizador podría parecer que ha convergido, pero en realidad es oscilante en un mínimo local, tiene la oportunidad de escapar de ella y que converja en el mínimo global o un mejor mínimo local. Por esta razón, el optimizador debe pasar un cierto número de iteraciones en torno a un punto mínimo antes de considerar que converge.

3. Relativa error cuadrático medio: este criterio es similar a la del umbral de transformación relativa, mediante el error cuadrático medio en lugar de los incrementos de

rotación/traslación.

4. Error cuadrado medio Absoluto: se detiene las iteraciones cuando el error entre las dos nubes alineadas está por debajo de un cierto valor.

Todas estas técnicas mantienen una relación entre la calidad y el tiempo de ejecución del procedimiento de registro. Lo que ha demostrado ser esencial para los algoritmos de optimización de rendimiento en tiempo real.

Debido a la presencia de numerosos mínimos locales en la función de error ICP, se puede realizar una serie de controles al final del proceso de registro, con el fin de confirmar una convergencia exitosa.

En el caso de sensores que obedecen al modelo de cámara pin-hole (RGB-D), esto se puede hacer de manera eficiente en espacio de la imagen por la prestación de un nube de puntos en la parte superior de la otra con las poses calculadas. Del mismo modo, se puede comprobar si la zona de solapamiento tiene suficientes inliers. Otro indicador del éxito de la inscripción es el número de condición de la zona de solapamiento, que indica si la superficie común de las nubes de puntos es lo suficientemente estable como para que no se permita el deslizamiento.

5.8 Transformación de cambio de base

El problema que se plantea para registrar o solapar dos nubes de puntos, es cambiar el punto de vista o sistema de referencia de una de las nubes fuente al sistema de otra nube objetivo. Las transformación se dividen en traslación para movimiento lineal y rotación sobre cualquiera de los tres ejes.

Una transformación homogénea en tres dimensiones es realizada mediante una matriz de transformación afín 4×4. Esta matriz se usa para proyectar cada punto en el espacio cartesiano con respecto a un punto de vista específico. A continuación, tenemos que $\mathbf{v_1} = (x_1, y_1, z_1, 1)^T$ sea un punto cuya base está definida por el punto de vista inicial y sea $\mathbf{v_2} = (x_2, y_2, z_2, 1)^T$ un punto cuya base está definida por el punto de vista cualquiera diferente al inicial. Entonces es posible expresar v_2 respecto a la base del punto de vista inicial como $T\ \mathbf{v_1} = \mathbf{v_2}$, donde T es una matriz de transformación afín definida:

$$T = \begin{pmatrix} r_{1,1} & r_{1,2} & r_{1,3} & t_{1,4} \\ r_{2,1} & r_{2,2} & r_{2,3} & t_{2,4} \\ r_{3,1} & r_{3,2} & r_{3,3} & t_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{pmatrix} \tag{5.7}$$

La matriz de transformación en la ecuación 5.7 se representa una transformación afín si $a_{4,1}=a_{4,2} = a_{4,3} = 0$ y $a_{4,4} \neq 0$. Las transformaciones afines se construyen con una matriz de rotación **R** de dimensión 3 × 3 y vector columna **t** representa una traslación. En la siguiente figura se ilustra la idea del problema planteado.

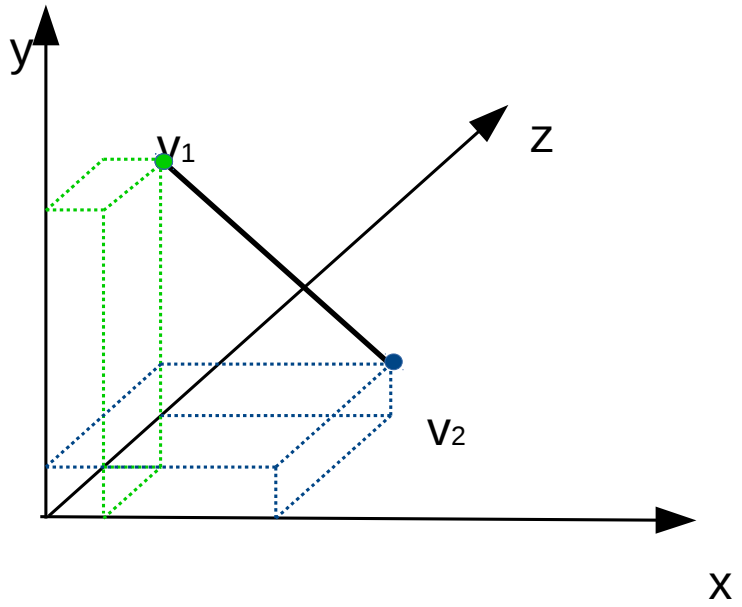


Figura 5.5: desplazamientos de puntos en transformación homogénea.

También existe la posibilidad de calcular los movimientos mediante ángulos de Euler, los cuales vienen representado en la figura 5.6, estos son dados dos sistemas de coordenadas *xyz* y *XYZ* con origen común, es posible especificar la posición de un sistema en términos del otro usando tres ángulos α, β y γ.

La definición matemática es estática y se basa en escoger dos planos, uno en el sistema de referencia y otro en el triedro rotado. En el esquema de la figura 5.6 serían los planos *xy* y *XY*. Escogiendo otros planos se obtendrían distintas convenciones alternativas, las cuales se llaman de Tait-Bryan cuando los planos de referencia son no-homogéneos (por ejemplo xy y XY son homogéneos, mientras xy y XZ no lo son).

La intersección de los planos coordenados *xy* y *XY* escogidos se llama línea de nodos, y se usa para definir los tres ángulos:

1. α es el ángulo entre el eje *x* y la línea de nodos.
2. β es el ángulo entre el eje *z* y el eje *Z*.
3. γ es el ángulo entre la línea de nodos y el eje *X*.

Los tres ángulos de Euler descritos son los valores de las tres rotaciones intrínsecas que describen el sistema, que son precesión, nutación y rotación. Estos valores servirán para determinar movimientos no lineales, como se producen en los sólidos rígidos al realizar giros.

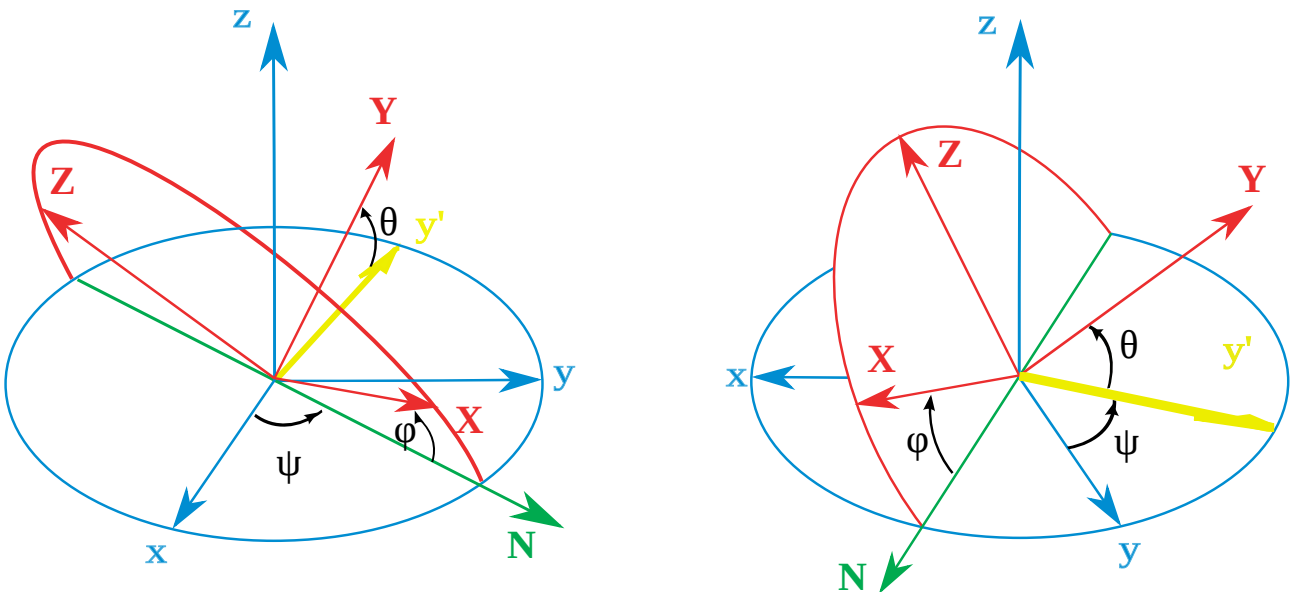


Figura 5.6: representación de las transformaciones mediante ángulos de Euler

5.9 Clustering y Segmentación

La agrupación y procesamiento de grandes nubes de puntos es uno de los principales cuellos de botella en los sistemas de percepción 3D. Aunque los algoritmos que podemos escribir para computar conjuntos de pequeñas nubes de puntos necesitan poca capacidad computacional, sus capacidades están limitadas para los grandes conjuntos de datos que pueden acumularse. Un ejemplo en este sentido, puede ser el problema de conseguir la mejor aproximación al plano de un conjunto de puntos. Sin entrar en los detalles matemáticos, es suficiente decir que el mejor modelo computacional es midiendo las distancias de todos los puntos pertenecientes al conjunto y compararlos, obteniendo así una estimación del plano que mejor se ajusta. Esto, evidentemente será más rápido para procesar una nube de puntos cuanto menor sea el conjunto de datos que tiene. La idea es crear un mecanismo que sea capaz de manejar grandes volúmenes de datos, dividiéndolos en cantidades menores, para acelerar el procesamiento. Sin que se produzcan grandes desviaciones del resultado esperado. Los resultados de los conjuntos truncados tienen que tener dos reglas, la primera es que permita a los datos ser procesados más rápido y la segunda es que permita el agrupamiento con alguna propiedad (color, curvatura,..). Esta última regla es de gran importancia para el desarrollo de algoritmos de procesamiento en 3D.

La segmentación de conjunto de datos en sí mismo es la búsqueda de ciertos patrones o estructuras en el conjunto de nubes de puntos con resultados de cierta calidad, esta estará restringida por :

1. La complejidad de las estructuras buscadas.

2. Los niveles de ruidos en los datos.

Ambos parámetros influyen en las propiedades computacionales de la búsqueda. Para ahorrar tiempo, estos algoritmos se ajustan para ser incrustados con generadores de hipótesis heurísticas tales como RANSAC, necesita límites altos de iteraciones para obtener un resultado robusto.

En un mundo ideal, habría base de datos de modelos simplificados para todos los objetos del mundo, los cuales podrían sustituir a los puntos de la nube P por sus modelos subyacentes. Sin embargo, esto es difícil, aunque una de las técnicas de simplificación de datos es sustituir por modelos geométricos primitivos en 3D. En este contexto, nos referimos a geometrías primitivas, como aquellas que tienen características básicas y fáciles de computar como puede ser planos, cilindros, esferas, y otros modelos polinomiales.

Los modelos de trabajo más utilizados, son sobre superficies planas que ayudan a eliminar los puntos pertenecientes a estas, con su consecuente disminución de datos. Cuando empezamos a resolver un problema de segmentación será más fácil buscar inicialmente las superficies planas que pueden contener objetos como es el suelo o una mesa o de las paredes. Un método que acelera la búsqueda es el algoritmo RANSAC. Para buscar planos necesitamos 3 puntos no colineales, podemos explicar el algoritmo con los siguientes pasos:

1. Selección aleatoria de 3 puntos no colineales {pi,pj,pk}.

2. Busqueda de plano que sirva de modelo (Ax+By+Cz+D=0).

3. Cálculo de las distancias entre el plano modelo (A,B,C,D) y todos los puntos de la nube.

4. Contabilizar todos los puntos que la nube que cumplen la condición $0 \leq |d| \leq |dt|$, donde dt es el umbral especificado.

El último paso es una de las maneras de cuantificar el error que ofrece el plano modelo con los tres puntos no colineales elegidos aleatoriamente, es necesario repetir este proceso de forma iterativa, con diferentes puntos, y elegir aquellos modelos que establezcan un error mínimo.

El ejemplo es un resumen de los pasos computacionales que deben hacerse para encontrar un plano, sin embargo, es necesario aplicar otras restricciones geométricas para otras figuras.

5.9.1 Técnicas básicas de agrupamiento.

De acuerdo con los objetivos presentados anteriormente, un método de agrupamiento necesita dividir una nubes de puntos P desorganizada en partes pequeñas para que el tiempo total de procesamiento se reduzca significativamente. La mayoría de los métodos simples en esta categoría

se centran en técnicas de descomposición espacial, que buscan subdivisiones y límites para permitir que los datos se agrupen en base a una medida determinada. Esta es usualmente representada como norma Minkowski, con las instancias más populares son las distancias Manhattan(L1) y Euclídea(L2).

Una agrupación de datos simple enfocado en sentido Euclidiano puede ser implementado haciendo uso de subdivisiones de cuadrícula 3D de espacio con ancho fijo, o más general, una estructura de datos de árbol. Esta representación particular es muy rápida de construir y es útil para situaciones donde ya hay una representación volumétrica de espacio ocupado. Los resultados de cada caja 3D(o estructura de árbol) pueden ser aproximados con una estructura diferente.

El método de cuadrícula 3D, solo tiene sentido, en aplicaciones que requieren subdivisiones iguales. Situaciones donde los agrupamientos pueden tener tamaños diferentes, nosotros necesitamos un algoritmo más complejo. Para lograr este objetivo, el sistema necesita entender primero que es un “object point cluster”y las diferencias con los puntos de otros agrupamientos. En un sentido más matemático, un agrupamiento de o cluster es definido como. Si tenemos $O_i = \{ p_i \in P \}$ que es un grupo de puntos distintos debe $O_j = \{p_j \in P \}$ si:

$$\min ||p_i - p_j||_2 \geq d_{th} \tag{5.8}$$

donde d_{th} es el umbral de distancia máxima.

En las ecuaciones de estado anteriores, si la distancia mínima entre conjuntos de puntos $p_i \in P$ y otro conjunto $p_j \in P$, es superior que la distancia dada por el valor, entonces los puntos en p_i pertenecen a los puntos del cluster O_i y los que están en p_j a otro cluster O_j distinto. Desde el punto de vista de la implementación, es importante para tener una noción sobre cómo es esta distancia mínima entre dos conjuntos que pueden ser estimados. Una solución es hacer uso de aproximaciones de consulta de vecinos más cercanos, a través de la representación kd-tree, el algoritmo que necesita podría tener los siguientes pasos:

1. Crear una representación kd-tree para la entrada de datos de la nube de puntos P.

2. Configurar una lista vacía de cluster C, y una cola de los puntos que necesita ser revisado Q.

3. A continuación por cada punto $p_i \in P$ realice los siguientes pasos:

    1.1. Agregar p_i a la cola actual Q.

    1.2. Para todos los puntos $p_i \in Q$ hacer:

- Buscar para el conjunto P_i^k de los puntos vecinos de p_i de en una esfera con radio $r < d_{th}$..

-Para todos los vecinos $p_i^k \in P_i^k$, comprobar si los puntos que están alrededor han sido procesados y sino añadirlos en Q.

1.3. Donde la lista de todos los puntos en Q han sido procesados, añadiendo Q a la lista de cluster C e inicializado a cero la lista Q.

4. El algoritmo termina cuando todos los puntos $p_i \in P$ han sido procesados y están ahora parte de ellos en la lista de puntos del cluster C.

Aplicado sobre el conjunto de datos de la nube de puntos, los modelos que quedan apoyados sobre el plano, el algoritmo propuesto construye un conjunto de agrupaciones de objetos separados euclideanamente.

5.9.2 Técnicas de orientación a los bordes en datos 3D

Junto con otras representaciones de características para puntos 3D, las estimaciones de curvatura en la superficie tiene una interesante característica que hacen adecuado para el propósito de segmentación. Porque se aproximan a la geometría de la superficie subyacente muestreada alrededor de un punto p_i, esto es útil para determinar aquellos puntos con una curvatura con valores extremadamente altos los cuales representa la geometría del borde para los datos de la nube de puntos P. Mientras el concepto de procesamiento de unos bordes en imágenes 2D es definido usando el gradiente de la imagen.,en escenas 3D podemos definir como lugares donde la geometría de la escena cambia abruptamente.

La umbralización de los puntos que tiene altos valores de curvatura puede ser realizada por el análisis de la distribución de curvaturas en una nube de puntos P. Las curvatura de la superficie pueden ser estimadas realizando un estudio de los valores propios de la matriz de covarianza alrededor de la zona del punto consulta p_q usando la siguiente ecuación:

$$\sigma_p = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2} \tag{5.9}$$

Una representación gráfica de las curvaturas en cada punto para el conjunto de datos, podemos ver las distribución de curvaturas en un histograma. Un punto puede ser considerar como parte de un borde ya sea sobre la base de un determinado umbral fijo o mediante la imposición de un o

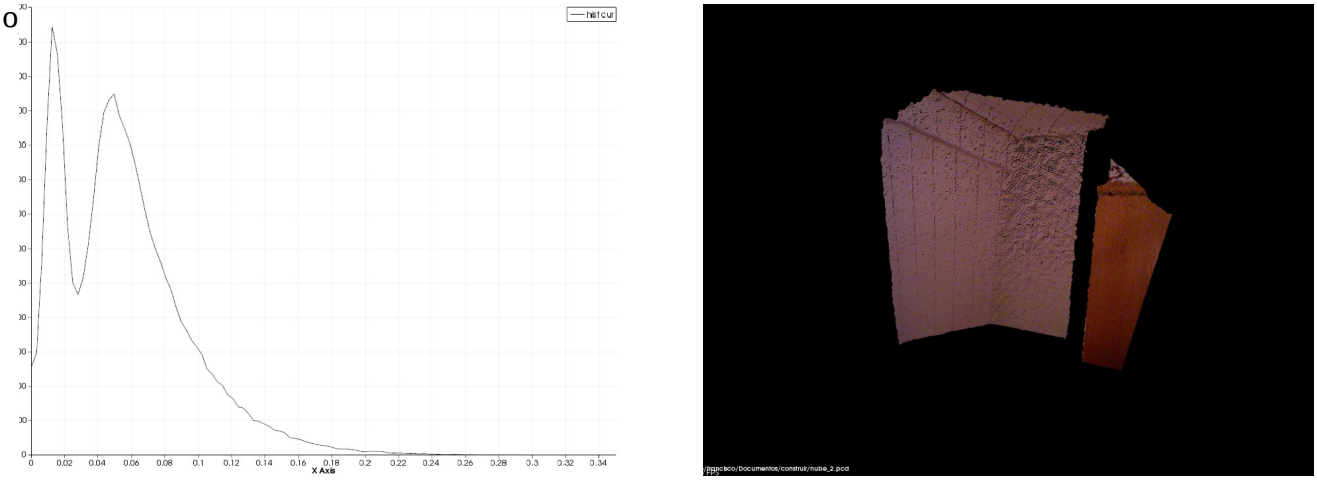


Figura 5.7:(derecha) Nube de puntos con planos de pared.(Izquierda)Histograma de curvaturas

Además de puntos con curvatura muy alta, la localización de los puntos sobre el límite geométrico de las superficie aparecen como bordes. La definición de un *punto límite* puede ser dada por el análisis de los ángulos que forman con respecto a sus vecinos. Obtenemos $\Theta = \{\theta_1 \cdots \theta_n\}$ es la lista de ángulos ordenados entre las líneas por dos puntos p_{k1} y p_{k2} alrededor de P^k con los puntos de consultas p_q. Entonces p_q es un punto localizado en el límite de una superficie, si:

$$\max(\alpha = \theta_{i+1} - \theta_i) \geq \alpha_{th} \tag{5.10}$$

donde α_{th} es el umbral del ángulo máximo, con un valor de $\alpha_{th} = \pi/2$ se obtienen buenos resultados en muchos casos.

5.9.3 Segmentación mediante Región de Crecimiento

La prolongación natural del los algoritmos de agrupamiento euclidiano propuesto en la sección de técnicas básicas de agrupamiento para el problema de segmentación de puntos con similares propiedades, es incluir información de los parámetros , como el color de los puntos o otra información relacionada con la geometría circundante.

Un ejemplo será: buscar y conectar regiones segmentadas de P con superficies lisas que están separadas por borde y cambios repentinos en la curvatura y de la orientación de la normal. El algoritmo empezaría agregando un punto $p \in P$ con valores pequeños de curvatura en la cola (inicialmente vacía), y verificar cada uno de estos p_k vecinos para ver si este podría permanecer en la misma región que p. La base del criterio en el cual un vecino p_k se dice que pertenece a una misma región:

$$\arccos(\langle \vec{n}, \vec{n}_k \rangle) \leq \alpha_{th} \tag{5.11}$$

donde $\vec{n}$ y $\vec{n}_k$ son las normales de la superficies en los puntos p y p_k. Si el ángulo formado por las normales no excede al dado por el umbral α_{th}, el punto p_k puede ser agregado a la región que pertenece p. Como verificación adicional se puede realizar sobre la estimación de la curvatura de la superficie, la cual es necesaria para los límites predefinidos por la curvatura σ_{th}. Si p_k cumple con la verificación extra como buena, se agrega a la cola de la lista, y procedemos a restaurarla.

Este método es similar para un enfoque de región de crecimiento. Con la diferencia de que el

crecimiento se propaga a lo largo de las direcciones de menor curvatura. Los puntos que dejan de ser añadidos a la región actual si ninguno de sus vecinos cumple con el criterio del ángulo y la cola de puntos está vacía(significa que todos los puntos con pequeña curvatura ya ha sido considerados). El algoritmo puede ser extendido para calcular automáticamente una buena relación de calidad de σ_{th}.

5.10 RANSAC

En gran medida, el análisis de las escenas (y, de hecho, la ciencia en general) tienen que ver con la interpretación de los datos obtenidos en términos de un conjunto de modelos predefinidos. Conceptualmente, la interpretación implica dos actividades diferentes. En primer lugar, existe el problema de encontrar la mejor coincidencia entre los datos y uno de los modelos disponibles (el problema de clasificación), en segundo lugar, existe el problema de calcular los mejores valores para los parámetros libres del modelo seleccionado (el problema de la estimación de parámetros). En la práctica, estos dos problemas no son independientes, una solución de la clasificación estimación de parámetros se requiere a menudo para resolver el problema de estimación .

Hay técnicas clásicas para la estimación de parámetros, como los mínimos cuadrados, optimizar (de acuerdo a una función objetivo especificado) el ajuste de una descripción funcional (modelo) para todos los datos presentados. Estas técnicas no tienen mecanismos internos para detectar y rechazar errores manifiestos. Se basan en la suposición de la desviación máxima (la suposición de suavizado) esperada de cualquier dato desde el modelo asumido, es una función directa del tamaño del conjunto de datos, independientemente de su tamaño el conjunto tendrá siempre suficientes valores buenos para suavizar cualquier desviación brutos.

En muchos problemas de estimación de parámetros prácticos el supuesto de suavizado no se sostiene; los datos contienen errores graves no compensados. Para hacer frente a esta situación, se han propuesto varias heurísticas. La técnica generalmente empleada se utiliza todos los datos para derivar los parámetros del modelo, a continuación, localizar el dato que está más alejado de acuerdo con el modelo instanciado, asumiendo que es un grave error, eliminar, e iterar este proceso hasta que o bien la desviación máxima es inferior a un umbral preestablecido o hasta que no existan suficientes datos para proceder.

El paradigma RANSAC, es capaz de suavizar los datos que contienen un porcentaje significativo de errores flagrantes. Este paradigma es particularmente aplicable a análisis de la escena porque los detectores de características locales, que a menudo cometen errores, son la fuente de los datos facilitados a los algoritmos de interpretación. Los detectores de características locales producen dos tipos de errores - errores de clasificación y los errores de medición. Errores de clasificación se obtienen cuando un detector de función identifica de forma incorrecta una porción de una imagen como una ocurrencia de una característica. Los errores de medición ocurren cuando

el detector función identifica correctamente la función, pero calcula mal uno de sus parámetros (por ejemplo, la ubicación de la imagen). Los errores de medición en general siguen una distribución normal, por lo que la hipótesis de suavizado es aplicable a ellas. Errores de clasificación, son graves, teniendo un efecto significativamente mayor al de errores de medición, los cuales no se promedian.

El procedimiento RANSAC es opuesto al de las técnicas convencionales de suavizado. En lugar de utilizar la mayor cantidad de los datos como sea posible para obtener una solución inicial, luego intentar eliminar los puntos de datos no válidos, RANSAC usa un conjunto de una cantidad inicial pequeña de datos que sean factible y cuando es posible lo agranda con datos consistentes. Por ejemplo, dada la tarea de encajar un plano para un conjunto de puntos tridimensionales, el enfoque RANSAC selecciona un conjunto de tres puntos (puesto que se requiere tres puntos para determinar un plano), calcula los parámetros del plano y contabiliza el número de puntos que están lo suficientemente cerca del plano para sugerir su compatibilidad con el plano propuesto, sus desviaciones son lo suficientemente pequeñas para ser errores de medición. Cuando existen suficientes puntos compatibles, es posible emplear RANSAC una técnica de cálculo mediante mínimos cuadrados para calcular los parámetros, se realiza un alisado para la mejora de los parámetros con un conjunto de puntos mutuamente coherentes identificado. El método RANSAC es descrito formalmente como sigue:

> Dado un modelo que requiere un mínimo de n puntos de datos para crear instancias de sus parámetros libres, y un conjunto de puntos de datos P tal que el número de puntos en P es mayor que n [# (P)≥ n], seleccionar al azar un subconjunto S_1 de puntos de n datos de P y una instancia del modelo. Utilice el modelo de instancia M_1 para determinar el subconjunto S_1* de puntos en P que están dentro de una cierta tolerancia de error de M_1. El conjunto S_1* se llama el conjunto de consenso de S_1.
>
> Si # (S_1*) es mayor que algún umbral t, que es una función de la estimación del número de errores brutos en P, utilizar S_1* para calcular (posiblemente utilizando mínimos cuadrados) un nuevo modelo M_1*.
>
> Si # (S_1*) es menor que t, seleccionar al azar a un nuevo subgrupo S_2 y repita el proceso anterior. Si, después de algún número predeterminado de ensayos, no se ha encontrado conjunto de consenso, ya sea resolver el modelo con el mayor conjunto consenso encontrado, o terminar en fracaso.

Hay dos mejores evidencias para el algoritmo anterior. En primer lugar, si hay un problema relacionado con fundamento para la selección de puntos para formar los conjuntos S, utilizar un proceso de selección determinista en lugar de al azar; segundo, una vez que se ha encontrado un conjunto adecuado de consenso S* y calculado su modelo M*, añadir nuevos puntos de P que son consistentes con M* para S* y calcular un nuevo modelo sobre la base de este conjunto más grande.

El algoritmo RANSAC contiene tres parámetros no especificados:

1. La tolerancia de error se utiliza para determinar si un punto es compatible con un modelo.

2. El número de subconjuntos para probar.

3. El umbral t, es el número de puntos compatibles utilizados para dar a entender que se ha encontrado el modelo correcto.

5.11 Algoritmo Iterative Closest Point

Cuando se necesita resolver el problema de registro de dos nubes de puntos no siempre es posible resolver el problema de forma directa, sino que se hace necesario resolverlo con métodos iterativos para obtener la matriz de transformación final. Encontrar una transformación, en el que un mismo punto del mundo capturado,en dos puntos de vistas y con referencias distintas, tome una transformación rígida, de tal forma que,se pueda referenciar desde un único sistema de coordenadas. La transformación debe cumplir:

$$\forall\, p_i \in P,\quad \exists q_j \in Q \;\mid\; \|T\, p_i - q_j\| = 0 \tag{5.12}$$

o

$$D(P,Q) = \iint_{\Omega} \|T\, p(u,v) - q(f(u,v), g(u,v))\|^2 \, du\, dv = 0 \tag{5.13}$$

donde p(u,v) ∈ P, q(u,v) ∈ Q, siendo P y Q dos vistas de una superficie, (u,v) ∈ ℜ x ℜ son los parámetros del espacio para P y Q, f y g son las funciones de correspondencias (que significa p(u,v) y g(f(u,v),g(u,v)) representan el mismo punto de la superficie), Tp_i es el resultado de la aplicación de T a p_i y Ω es la región de superposición de P y Q. Hay que tener en cuenta que T es una matriz que realiza las operaciones de rotación y traslación de los puntos a transformar, si se tiene en cuenta que una rotación simple sobre un eje la matriz que obtendremos si fuera sobre el eje x:

$$R_x, \alpha = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -sen(\alpha) & 0 \\ 0 & sen(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \approx \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & -\alpha & 0 \\ 0 & \alpha & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.14}$$

haciendo la misma simplificación para el resto de los ejes la matriz de transformación por rotaciones puras sería:

$$R \approx \begin{bmatrix} 1 & -\gamma & \beta & 0 \\ \gamma & 1 & -\alpha & 0 \\ -\beta & \alpha & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.15}$$

Sin embargo teniendo en cuenta que las rotaciones que se realizan no son infinitesimales, por

lo que no se deben realizar la simplificación. En este caso se expresa en coordenadas homogéneas donde la matriz de transformación general T se expresa de la siguiente manera :

$$T(\alpha,\beta,\gamma,t_x,t_y,t_z)=\begin{bmatrix} c\alpha c\beta & c\alpha s\beta s\gamma - s\alpha c\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma & t_x \\ s\alpha c\beta & s\alpha s\beta s\gamma + c\alpha c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma & t_y \\ -s\beta & c\beta s\gamma & c\beta c\gamma & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.16}$$

donde, sx o cx sustituye a sen(x) y cos(x) respectivamente. Siendo α, β y γ son los ángulos de giro alrededor de los ejes x, y, z respectivamente y tx, ty y tz son los desplazamientos lineales. Por lo que en general, la transformación T necesaria para los puntos de vistas del registro, tiene 6 grados de libertad. Por lo tanto, la tarea de registro es en realidad para buscar dicha transformación en el espacio de parámetros de transformación de manera que D (P, Q) en la ecuación 5.12 alcanza su mínimo, que se puede conseguir mediante la resolución de la siguiente conjunto de ecuaciones:

$$\frac{\partial D}{\partial \alpha}=0,\quad \frac{\partial D}{\partial \beta}=0,\quad \frac{\partial D}{\partial \gamma}=0,\quad \frac{\partial D}{\partial t_x}=0,\quad \frac{\partial D}{\partial t_y}=0,\quad \frac{\partial D}{\partial t_z}=0 \tag{5.17}$$

La dificultad es que el proceso no el lineal, por lo que se utilizan métodos iterativos. Por otra parte, D (P,Q) no puede ser convexa en general, y no hay ninguna garantía de alcanzar un mínimo global por un procedimiento iterativo. Por último, por lo general, no conocemos las funciones f y g.

Potmesil[11] usa una búsqueda heurística en el espacio de parámetros de transformación para que coincida con la superficie, pero el segundo problema mencionado anteriormente todavía existe. El enfoque se basa en la suposición de que una transformación aproximada entre dos puntos de vista se conoce de antemano, es decir, están aproximadamente registradas. El propósito de hacer esto es doble. En primer lugar, se argumenta que el objetivo del algoritmo de registro superficial es encontrar una transformación más fina y precisa entre las diferentes vista de una superficie del objeto y que la transformación inicial aproximada puede obtenerse mediante el cotejo de alto nivel o mediante el conocimiento de la geometría de la configuración de adquisición de datos. En segundo lugar, ya que estamos seguros de que hay un mínimo global para D (P,Q) se puede encontrar un buen punto de partida.

De acuerdo con la definición dada para registro superficial, si conocemos un conjunto de N pares de puntos correspondientes, llamados puntos de control, en dos vistas $p_i \in P$ y $q_j \in Q$, i = 1 ... N, podemos encontrar fácilmente por la transformación minimizando,

$$e=\sum_{i=1}^{N} \| T\, p_i - q_i \|^2 \tag{5.18}$$

cuando el conjunto de puntos tiene un tamaño superior a 3. Desafortunadamente, la información de correspondencia es difícil de obtener, especialmente para superficies no estructuradas. Otro camino

para solucionar este problema consiste en minimizar las entre los de una superficie a otra, este problema consistiría en

$$e=\sum_{i=0}^{N}\|Tp_i-q_j\|^2,\ \ con\quad q_j=q|\min_{q\in Q}\|Tp_i-q\| \tag{5.19}$$

Esto se desprende directamente de la ecuación inicial, en el caso de $\min_{q\in Q}\|Tp_i-q\|=0$ para todo i=1,...,N, la ecuación 5.19 será cero. Pero esto es muy difícil de implementar, ya que encontrar q_i es en sí un problema de optimización. Aproximación del algoritmo por un método iterativo se puede utilizar si conocemos una transformación T^0 inicial que acerque P en el registro con Q, en este caso, en cada iteración k, se utiliza el valor anterior T^{k-1} para encontrar q_j^k:

$$e^k=\sum_{i=1}^{N}\|T^k\,p_i-q_j^k\|^2\quad con\quad q_j^k=q\ |\ \min_{q\in Q}\|T^{k-1}\,p_i-q\| \tag{5.20}$$

Este enfoque se necesita para llevar a cabo la reducción al mínimo en una superficie digital para encontrar q_j^k, como suele ser el caso. Si utilizamos un punto de aproximación en lugar de la q_j^k se define en la ecuación 5.20, el problema se vuelve más fácil. Potmesil [11] ha utilizado la distancia entre la dirección normal a la primera superficie como una función de las evaluaciones de registro. Siguiendo esta idea, una estimacion del error sería:

$$e^k=\sum_{i=1}^{N}\|T^k\,p_i-q'^k_j\|^2\quad con\quad q'^k_j=(T^{k-1}\,l_i)\cap Q \tag{5.21}$$

donde $l_i=\{a|(p_i\text{-}a)\times n_{pi}=0\}$ es la línea normal a P en p_i, n_{pi} es el vector unitario normal a la superficie de P en p_i y $(Tl_i)\cap Q$ representa el punto de intersección de la línea l_i(antes de ser transformada) con la superficie Q.

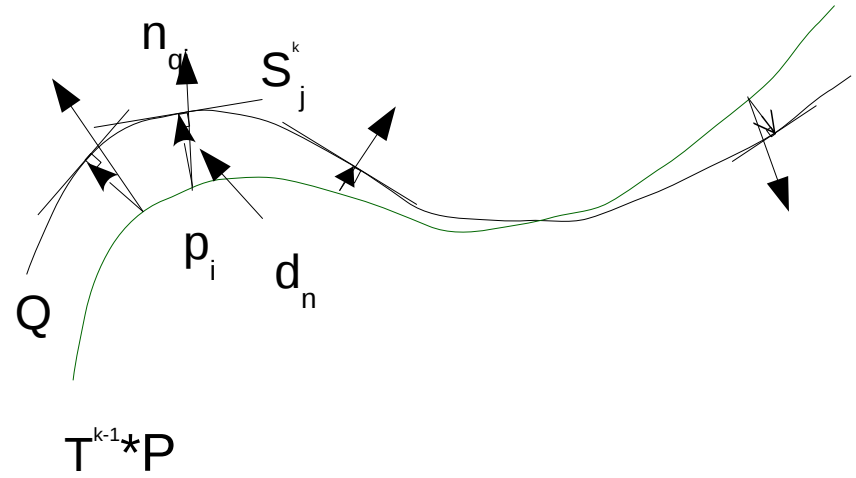


Figura 5.8: ejemplo visual de Iterative Closest Point

El problema común con los dos métodos iterativos anteriores es que, en cada iteración, se trabaja con algunos *puntos de control* que no necesariamente son verdaderos puntos de correspondencia. La consecuencia es una convergencia más lenta, ya que las restricciones impuestas

por los diferentes pares de puntos de control pueden ser incompatibles entre sí antes de que se haya obtenido el registro. El enfoque de la aproximación de Q usando su plano tangente de S_j a q_j de la ecuación 5.19, esta ecuación se puede escribir como:

$$e=\sum_{i=1}^{N}\|Tp_i-q'_j\|^2, \quad con \quad q'_j=q \mid \min_{q\in S_j}\|Tp_i-q\| \tag{5.22}$$

donde S_j es el plano tangente de Q en q_j. Como se ha mencionado anteriormente, se desconoce la transformación de q_j. Pero si tenemos una inicial T^0 como se ha mencionado anteriormente, entonces podemos empezar un proceso iterativo como una aproximación. En este caso se puede usar el q'^k_j definido en la ecuación 5.21 como una aproximación de q_j para la iteración.

Desde la distancia del punto al plano podemos expresar como una función lineal de las coordenadas del punto, la iteración producida puede ser formulada de la siguiente forma (ver figura 5.8):

$$e^k=\sum_{i=1}^{N} d_s^2(T^k p_i, S_j^k) \tag{5.23}$$

con

$$S_j^k=\{s|n_{qj}^k\cdot(q'^k_j-s)=0\}, q'^k_j=(T^{k-1}l_i)\cap Q \tag{5.24}$$

donde d_s es la distancia con signo desde un punto a un plano y n^k_{qj} es el vector normal a la superficie de Q en q'^k_j. Ahora no se trabaja con puntos de correspondencia específicos, no existiendo el problema de la convergencia más lenta. De hecho, se reduce al mínimo la distancia de un punto a un plano, que solo limita la dirección en la que esta distancia se puede reducir. El punto tiene otros dos grados de libertad, en el que se puede mover de acuerdo con las limitaciones impuestas por otros puntos y planos. Por lo tanto, la optimización global (en este caso, la minimización de la suma de las distancias) se puede lograr más rápidamente.

6 Cartografía de entornos interiores

Alguno de los dispositivos que realizan capturas de nubes de puntos son capaces de extraer, en un campo visual de 360º, la realidad que le rodea. Como los sensores más económicos tiene un campo visual más limitado es necesario realizar un registro de nubes. En este capítulo explicaremos los pasos necesarios para realizar la reconstrucción de una imagen 3D, enfocado para un hardware asequible que nos permita la construcción de un sistema de visión barato y robusto, en las zonas donde es capaz de operar.

En los siguientes apartados nos centraremos en los pasos necesarios para realizar un proceso de mapeo 3D mediante un sensor con un enfoque limitado. A partir de estudios previos realizados en el Anexo III, como el estudio de keypoints, analizaremos las propuestas que desde el blog de la librería PCL se recomienda para realizar un registros de imágenes. Por último, se compara con otro método a los propuestos por PCL, con la intención de atenuar los errores cometidos en los métodos de PCL.

6.1 Transformaciones Previas

Antes de empezar a realizar los procesos de análisis de la nube y obtener su información, es importante estudiar el sistema de referencia de la imagen. Es necesario establecer un origen, para solapar unas nubes con otras, que por sencillez tomaremos el convenio de tomar el punto y las direcciones de enfoque de la primera captura.

A cada nube le corresponde una matriz de transformación que está referida sobre la primera nube capturada. La posición real de los puntos de vista no son arbitrarias, lo que nos permitirá simplificar el registro de nubes, a realizar sobre pares de nubes a solapar. Si analizamos las transformaciones realizadas en cada captura a partir del esquema de la figura 6.1 donde R_{i-1} suponemos que le corresponde a la matriz unitaria, que no produce transformaciones, por ser la primera imagen capturada, denominada N_{i-1} como nube semilla. La primera imagen que se transforma será N_i, perteneciente a la segunda captura. La transformación que se realiza será en función de la nube N_{i-1}, siendo $R_i \neq I$ si existe movimiento.¡ para calcular R_i se aplica sobre la nube N_i y estas dos nubes estarán solapadas o registradas.

En la nube N_{i+1} existen dos anteriores, que se han transformado previamente. Para mantener un mismo criterio de restricción de distancias en los algoritmos de comparación, se realizará una transformación previa. La nube N_{i+1} se entrega como si estuviera posicionada en el origen. Desconocemos los parámetros de R_{i+1}, pero si es conocida una posición intermedia más cercana que es R_i. Por tanto, se realizará una pretransformación (transformación previa) $R'_{i+1}=R_i$, lo cual implicaría que se ha capturado la nube en la última que ya se ha registrado, estando en un caso

similar al primero. Para terminar de realizar el registro con las tareas necesarias para encontrar otra matriz de transformación R''_{i+1}, en la que se aplicará sobre la nube N'_{i+1}, que ha sido pretransformada. R''_{i+1} será la transformación que corresponde al movimiento entre las dos últimas imágenes. La nube registrada que llamaremos N^o_{i+1} por estar en referencia al origen, será obtenida de la siguiente manera:

$$N^o_{i+1} = N_{i+1} * R'_i * R''_{i+1} \tag{6.1}$$

Tenemos las transformaciones necesarias para realizar el registro y la pretransformación de la siguiente nube, que será:

$$R'_{i+2} = R_{i+1} = R'_i * R''_{i+1} \tag{6.2}$$

De esta forma queda establecido un mecanismo de aproximación, que facilitará registro de las nubes por cercanía de puntos. Redundará en mayor estabilidad del sistema de registro. Es posible establecer otras aproximaciones de tipo inercial, que ajustarán la nube en función del movimiento anterior.

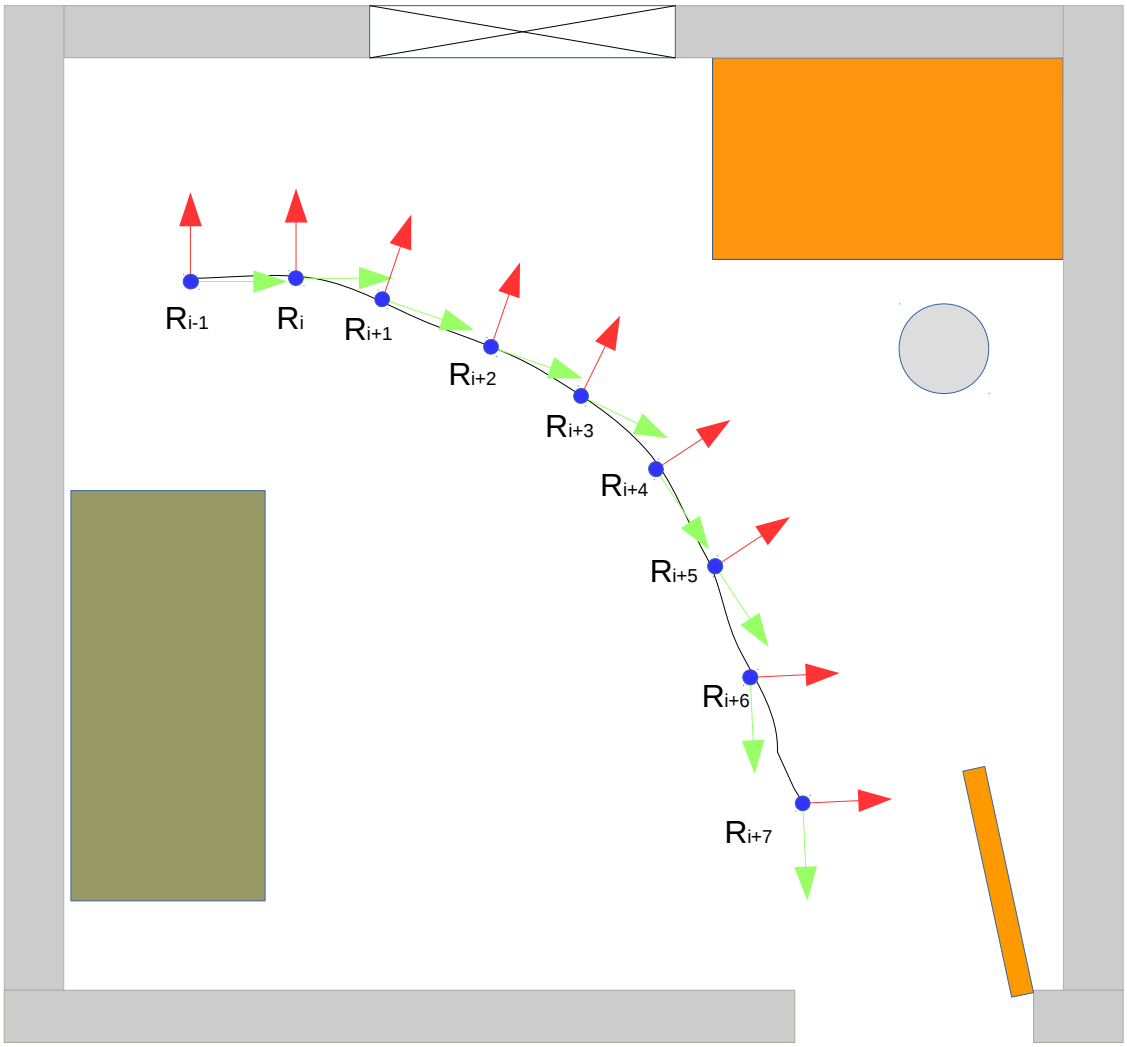


Figura 6.1: esquema de orientación y posicionamiento de captura de imágenes.

6.2 Propuesta de “Point Cloud Library”

Los desarrolladores de la librería PCL plantean un enfoque para resolver el problema de registros de nubes de puntos, el cual no ha sido avalado por ningún estudio que demuestre la validez del desarrollo. Aunque, la mayoría de los algoritmos utilizados si han sido estudiados, comprobando su funcionamiento y eficacia. Esto hace necesario comprobar si el conjunto de algoritmos funcionan para alcanzar el propósito.

La librería PCL ofrece una gran variedad de algoritmos que cumplen un mismo propósito, para el caso de registro de nubes, lo cual nos permite tener múltiples opciones para implementar un sistema de registro. El objetivo es implementar un método que sea capaz de resolver el problema en el menor tiempo posible y establecer cuales son los límites óptimos de trabajo.

Se estudiaran cuatro casos, en el primer caso se tomarán los algoritmos que mayor eficacia en tiempos de cómputo hemos comprobado, en los sucesivos se motivará las mejoras de calidad introducidas, con el objetivo de mejorar el resultado del registro. Para realizar el estudio se tomarán tres escenas diferentes en las que se capturaron nubes de puntos con desplazamientos controlados. Los movimientos a estudiar serán lineales, en dirección ortogonal al plano de captura del sensor, con pasos de desplazamientos de 10 mm con desplazamiento máximo de 300 mm y giros, en uno de los ejes paralelos al plano del sensor (en concreto el que está orientado hacia el plano más dominante), con ángulos de paso de 1º hasta un máximo de 10º. En las figura 6.2 se observa gráficamente los movimientos de muestreo que se realizan.

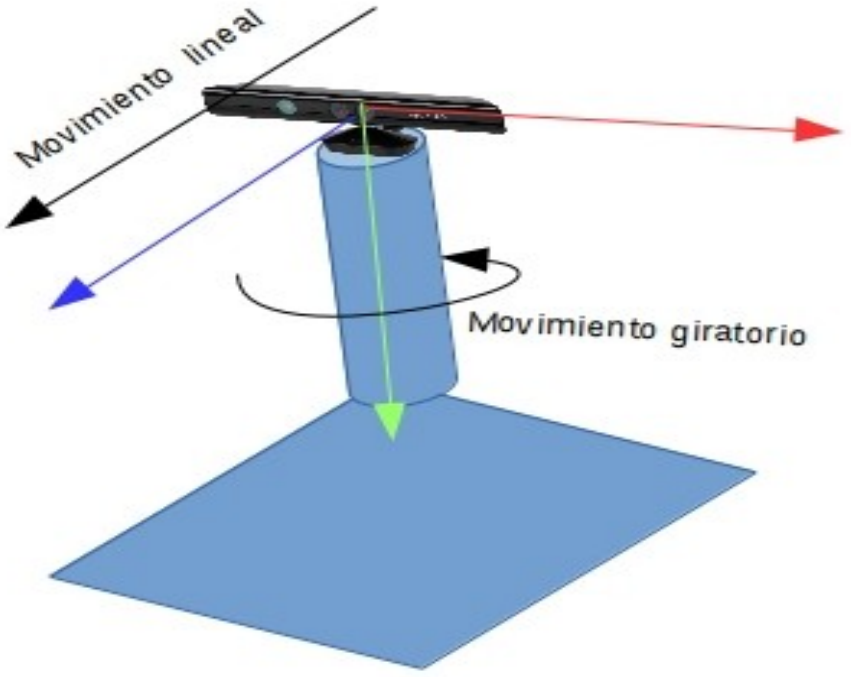


Figura 6.2: esquema de movimientos a estudiar

Las escenas serán capturadas en ambiente interior, mediante un sensor Kinect, con muestras de al menos 8 nubes que serán registradas cada una con todas las nubes de la muestra a estudiar. En el estudio que llevamos a cabo se tienen tres tipos de escenas, que son etiquetadas en la siguiente figura por el siguiente orden, habitación(a), cocina(b) y salón(c). Cada uno de los casos serán analizados por separado, con el fin de comprender cómo evoluciona para los diferentes movimientos, mostrando la media en cada paso y su desviación típica.

(a) (b) (c)

Figura 6.3: nubes de puntos estudiadas, (a) habitación, (b) cocina, (c) salón.

Para el análisis se tendrán en cuenta dos tipos de movimientos, que serán los más comunes que se produzcan si recurrimos a sistemas robotizados. Por un lado, un movimiento rectilíneo en el sentido del eje ortogonal al plano de captura del sensor, lo que equivale a caminar hacia delante. El otro movimiento es un giro de uno de los ejes no ortogonales al plano de captura del sensor. Por simplicidad, el movimiento de giro será equivalente a los desplazamientos en los sentidos paralelos al planos de captura del sensor. No se estudiará el giro sobre el eje ortogonal al plano de captura, si son generados, tienen poca amplitud y no redundará en una mayor amplitud visual que nos permita mapear. Para abordar este análisis se definirán tres tipos de errores que determinarán la calidad del algoritmo propuesto en cada caso.

El error medio ε_m, que se definirá como la distancia entre los centros de gravedad de la nube con aplicación de la transformación calculada al centro de gravedad de la nube con la transformación conocida. De tal forma que:

$$\varepsilon_m = P^c_{2\,cdg} - P^g_{2\,cdg} \tag{6.3}$$

donde P^{c}_{2cdg} es el centro de gravedad de la nube 2 después de aplicarle la transformación calculada y P^{g}_{2cdg} centro de gravedad de la nube 2 después de aplicarle la transformación conocida.

El error ponderado ε_p, es la media de todos los errores que se producen en la nube teniendo en cuenta todos sus puntos, de tal forma que:

$$\varepsilon_p = \frac{\sum_{i=1}^{n} p_{2i}^{c} - p_{2i}^{g}}{\sum_{i=1}^{n} i} \tag{6.4}$$

donde p^{c}_{2i} son los puntos de la nube 2 después de aplicarle la transformación calculada y p^{g}_{2} puntos de la nube 2 después de aplicarle la transformación conocida.

El error máximo será la distancia más grande que existe entre dos puntos que se pertenecen, de tal forma que:

$$\varepsilon_{max} = max\{\| p_i^c - p_i^g \| \ | p_i^c, p_i^g \in P_2\} \tag{6.5}$$

Los errores se han elegidos por la facilidad de cálculo cuando es conocido el movimiento, debido a que las correspondencias están predefinidas por los índices de la nube. Cada uno de estos errores servirán para analizar cómo se comporta el ajuste del solape de nubes de puntos. Si solo se expresara un tipo de error, no se podría analizar el comportamiento de los diferentes métodos.

Caso I

Este primer caso es el más sencillo, la imagen no necesita ser tratada con filtros. Se aplican procesos de matriz de imágenes 2D para calcular los keypoint. Los demás son básicos de la librería para el registro de nubes.

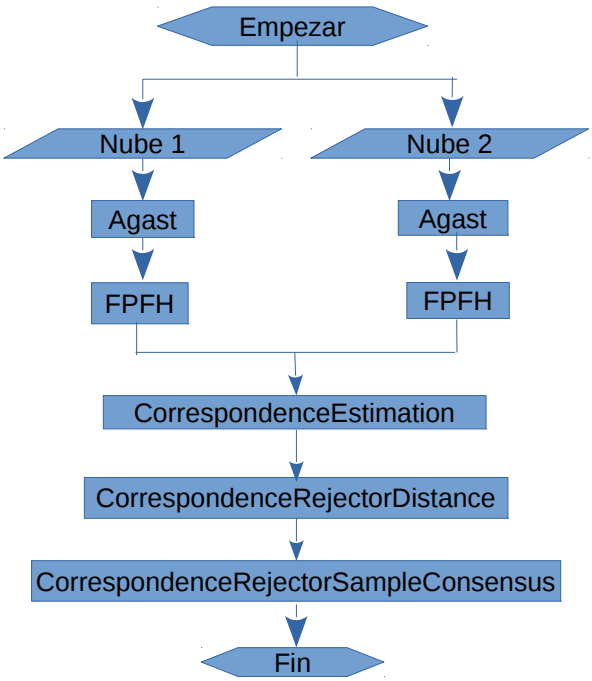
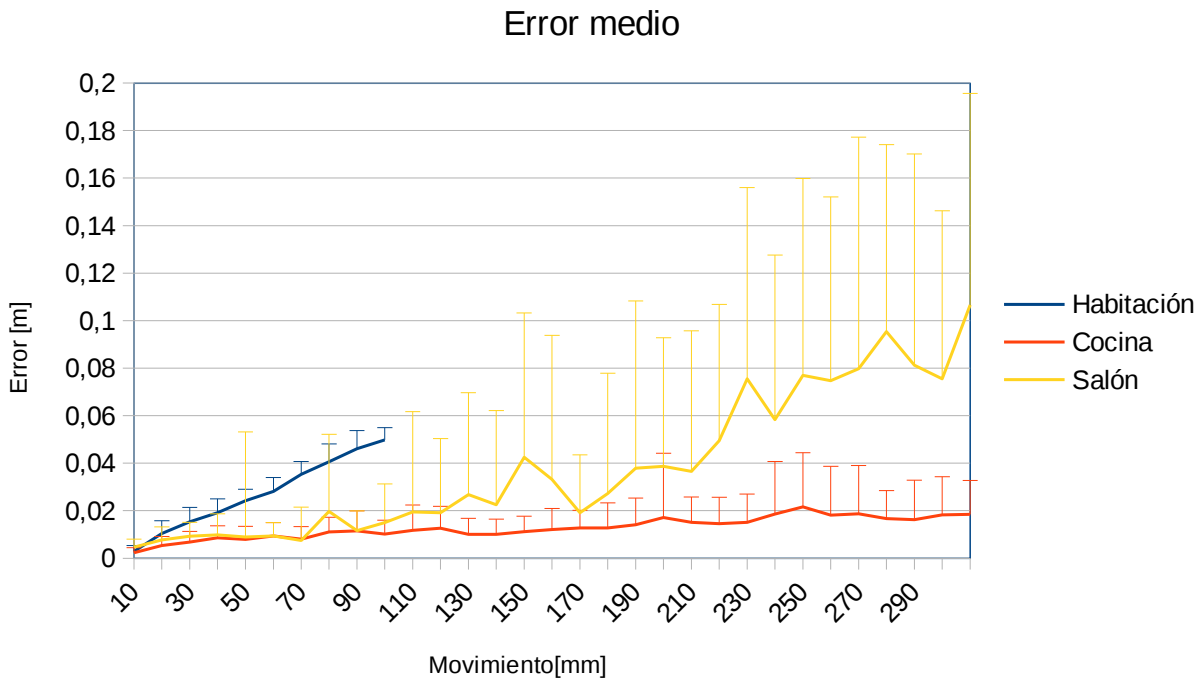
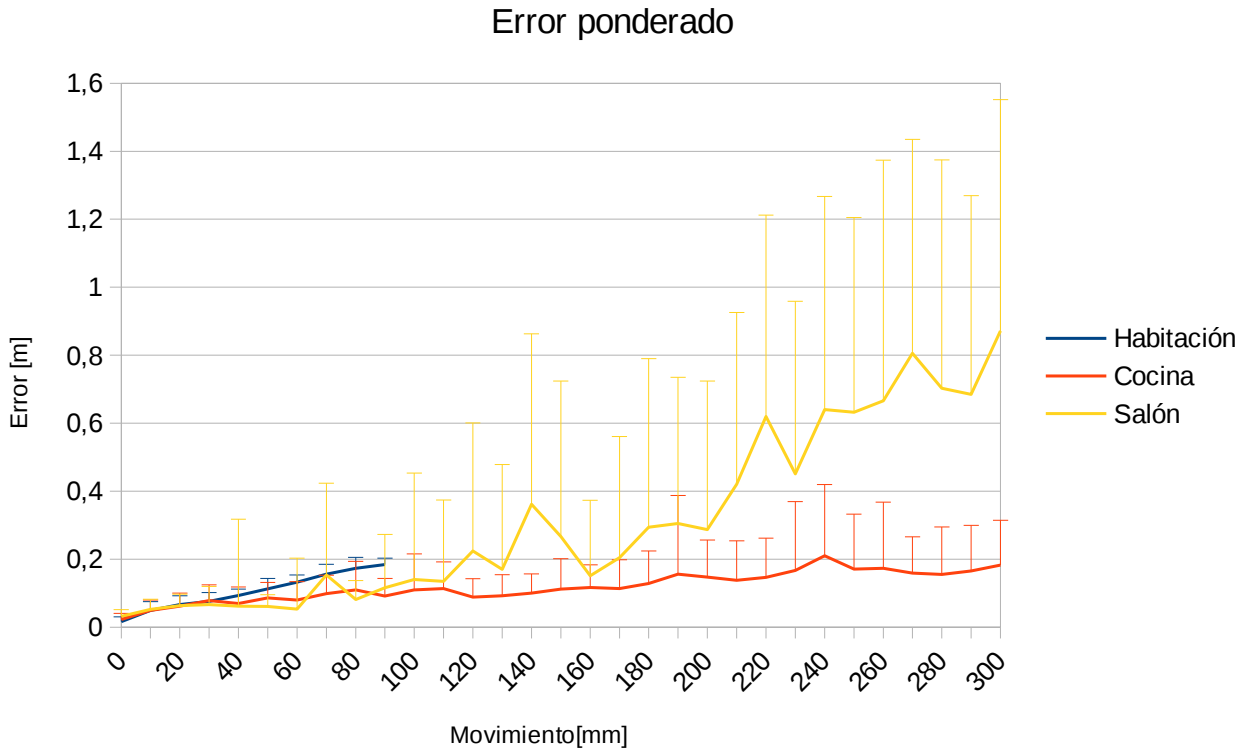


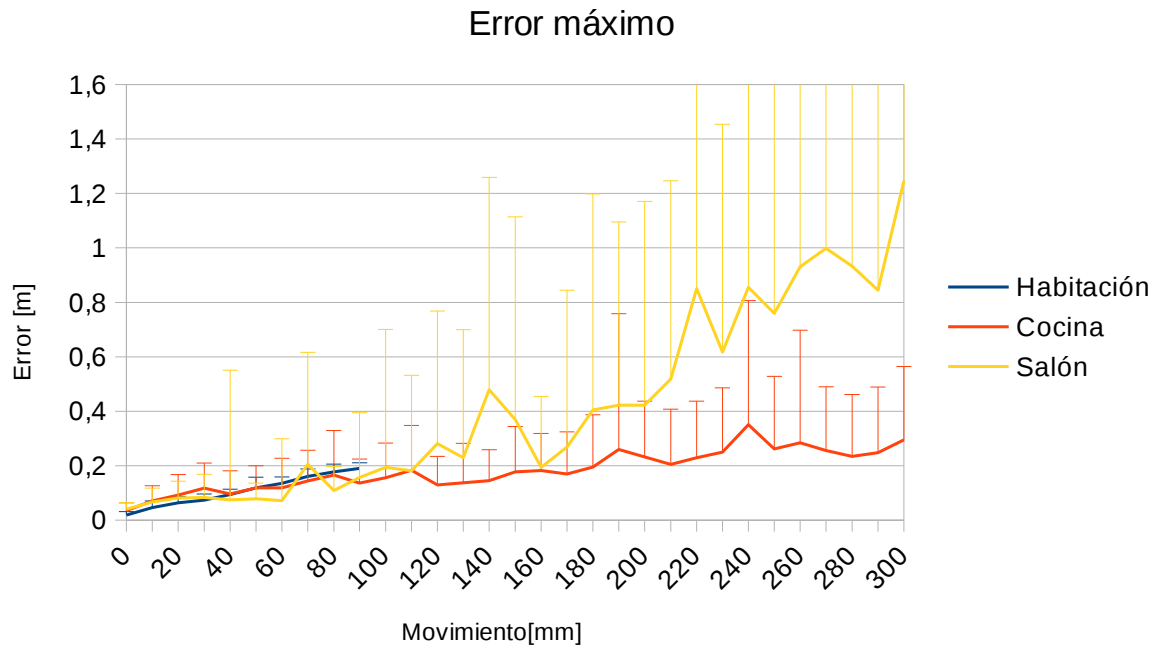
Figura 6.4: esquema de algoritmo para el caso I

En este caso se intenta que el proceso de solape sea lo más rápido posible, teniendo los tiempos de cómputo de los keypoints, que forma el verdadero cuello de botella en este proceso. En las siguientes gráficas se observa cómo se comporta frente a los errores, para determinar si es adecuado.

Gráficas de movimiento rectilíneo:



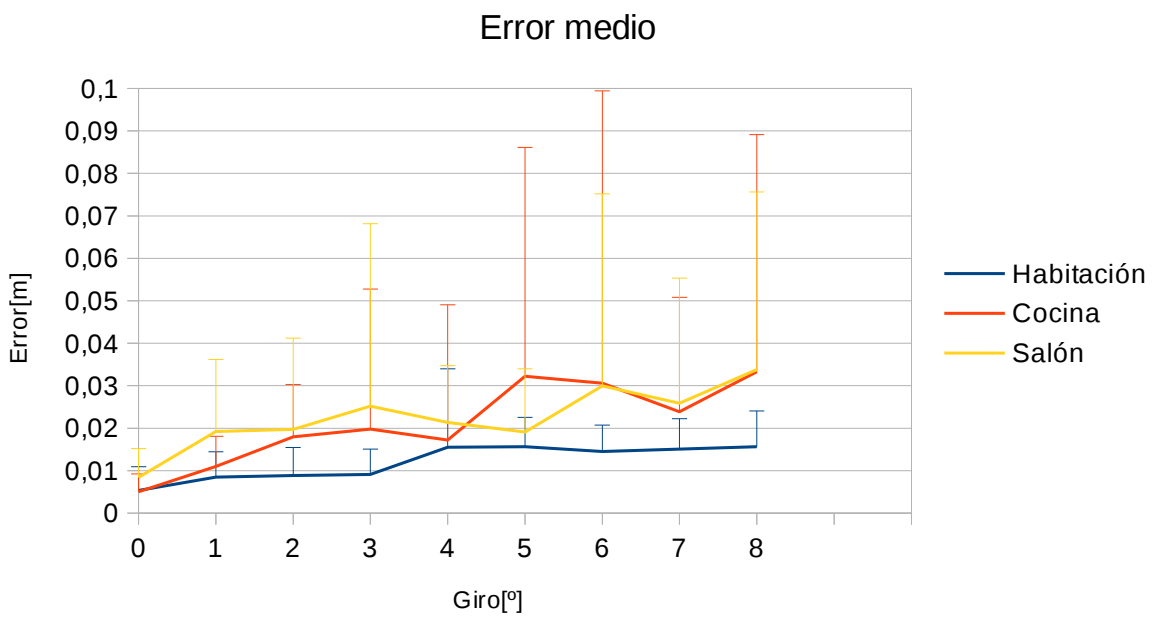


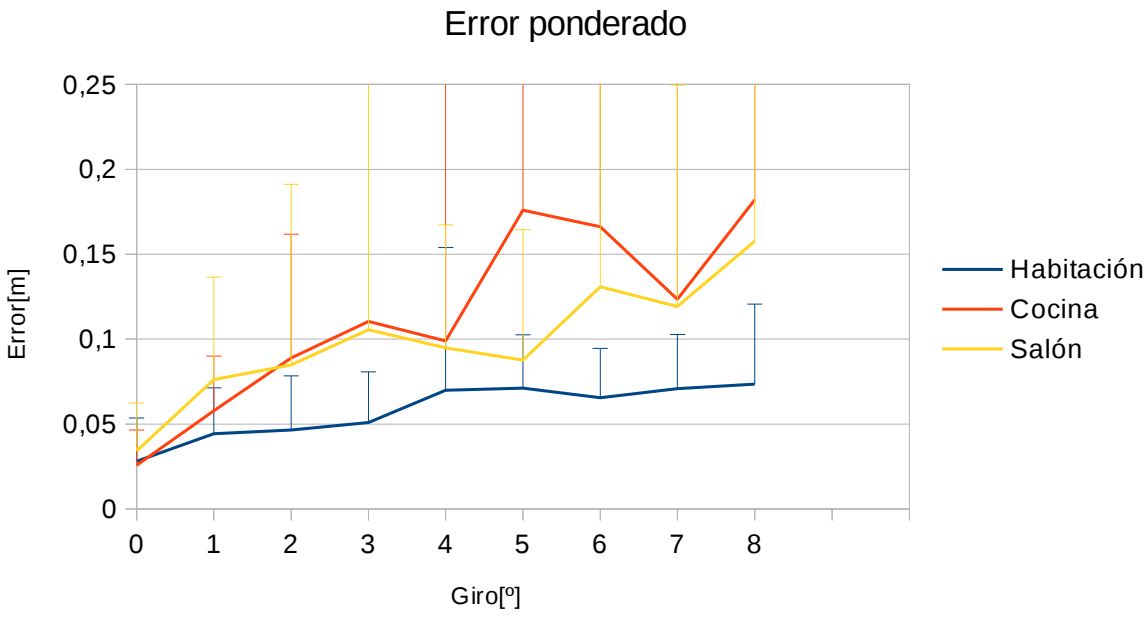


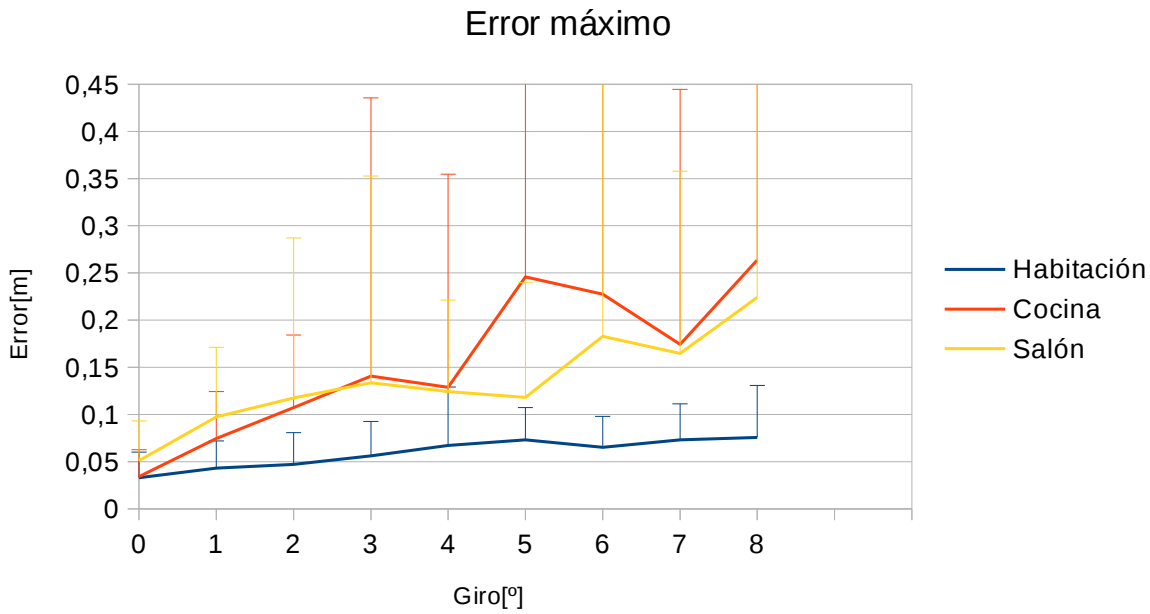
Los errores del registro para movimientos lineales son elevados, creciendo según es la distancia de movimiento a registrar. Son las desviaciones típicas las que indican la dispersión de los

resultados que se producen, de lo cual se deduce que en nubes con fuertes variaciones de planos es bastante inestable.

Gráficas de movimiento rotación:







En el caso de los giros existe mayor divergencia para ángulos pequeños, con errores ponderados y máximos muy elevados, si bien los el error medio es menor, por lo que es de suponer que no completa el giro exacto para lograr el solape adecuado.

En ambos movimientos existen errores elevados, lo cual dificulta un solape correcto. El error medio no es elevado con respecto al error ponderado y el máximo, esto quiere decir que se logra centrar la nube de forma adecuada, pero existen giros, que aunque sean pequeños, generan en los entornos de la nube errores grandes. De forma adicional, teniendo en cuenta la desviación típica, existe una variabilidad de resultados muy alta, lo cual el método no tiene convergencia en sus resultados. Todo esto hace que sea difícil la utilidad para el registro de nubes de puntos de forma consistente.

Caso II

Este caso es similar al anterior, la obtención de keypoints se realizará con un algoritmo de enfoque 3D, que genera menos cantidad de puntos. Esto supondrá un mayor consumo de tiempo en la etapa de obtención de keypoint, que se intenta atenuar con el filtrado de la nube. El objetivo de estos cambios es tener una menor cantidad de puntos, generados mediantes algoritmos propios para nubes de puntos, con los que debe ser más fácil obtener una correspondencia más fiable.

Como las normales y los descriptores utilizan la clases pcl::NormalEstimationOMP y pcl::FPFHEstimationOMP respectivamente, los tiempos de ejecución hay que dividirlos entre el número de hilos.

Se muestra en el siguiente diagrama las etapas de procesamiento:

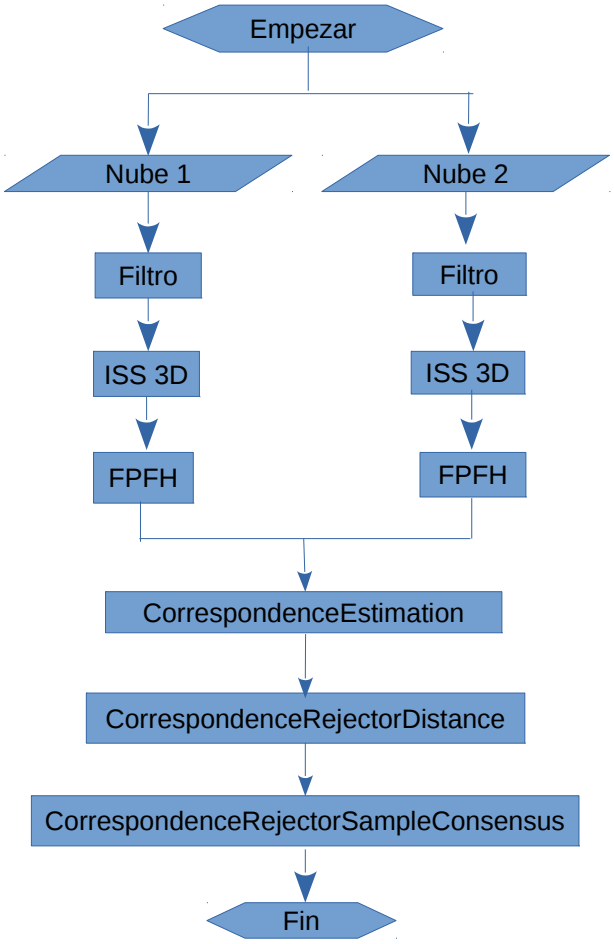
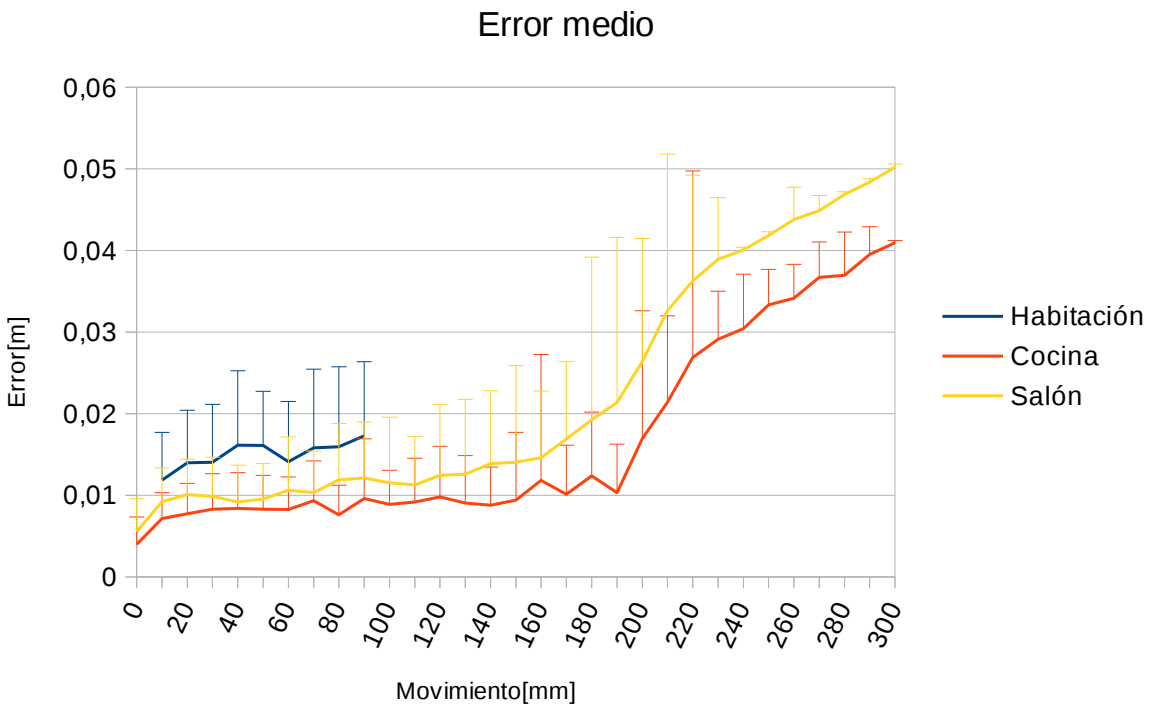
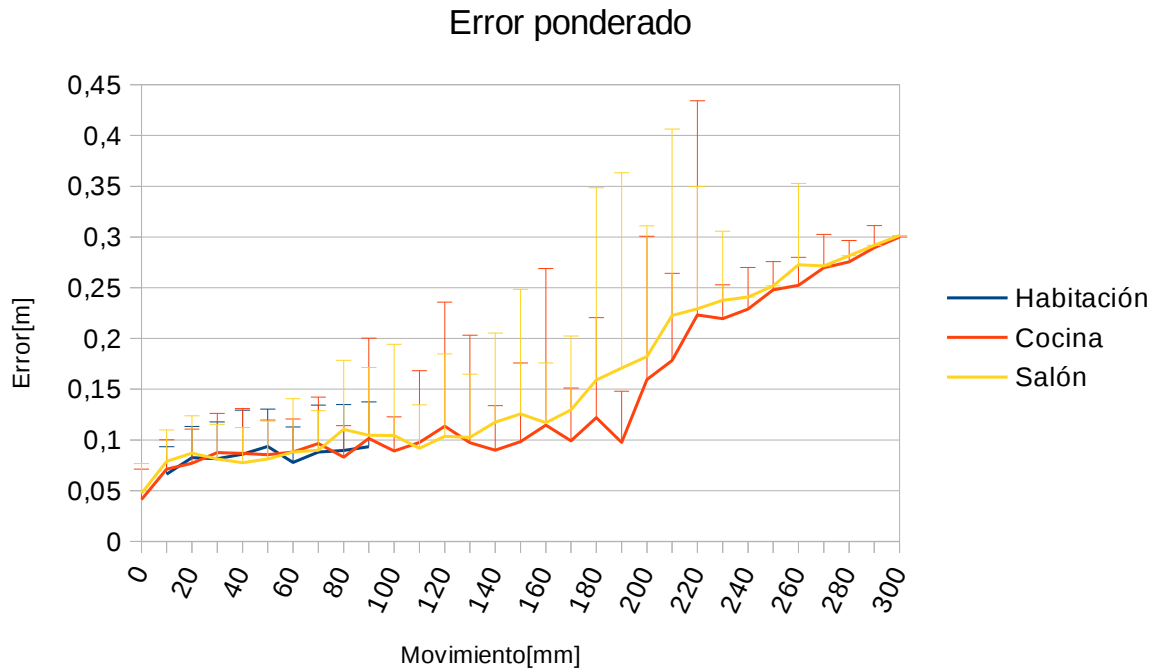
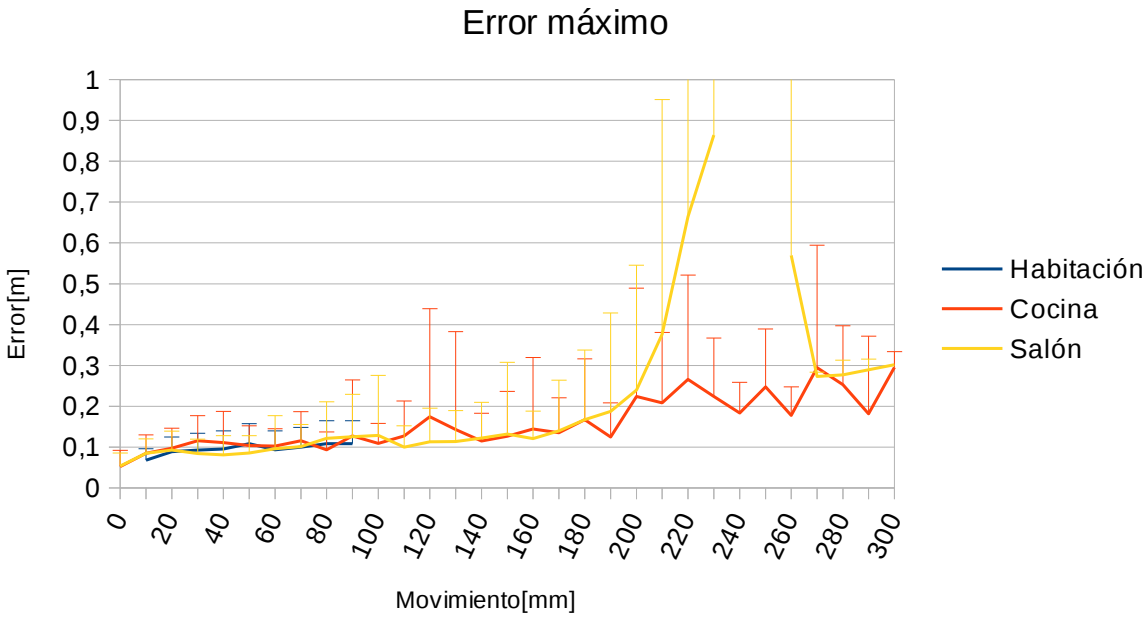


Figura 6.5: esquema de algoritmo para el caso II

Gráficas de movimiento rectilíneo;

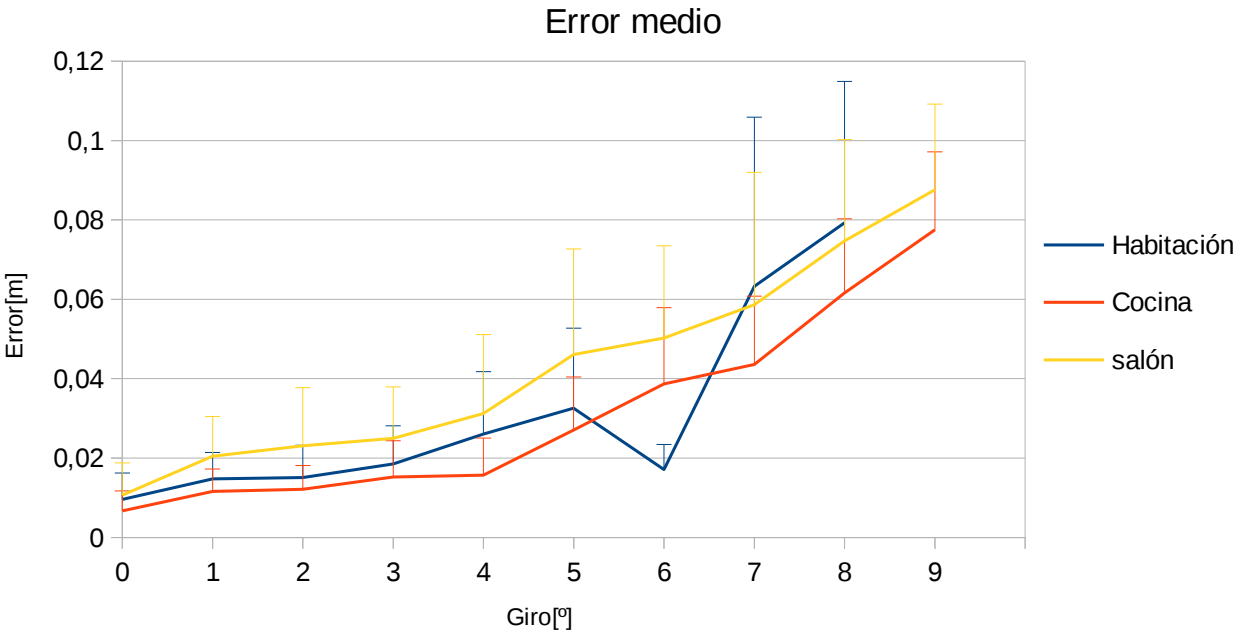


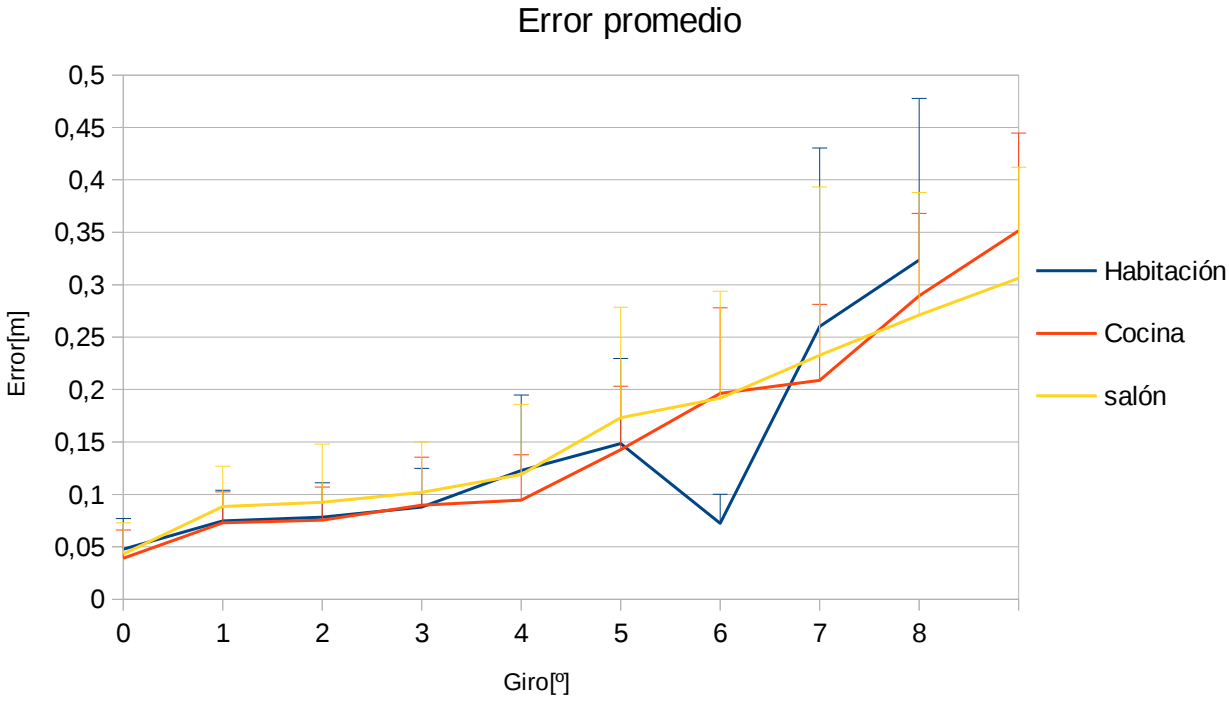


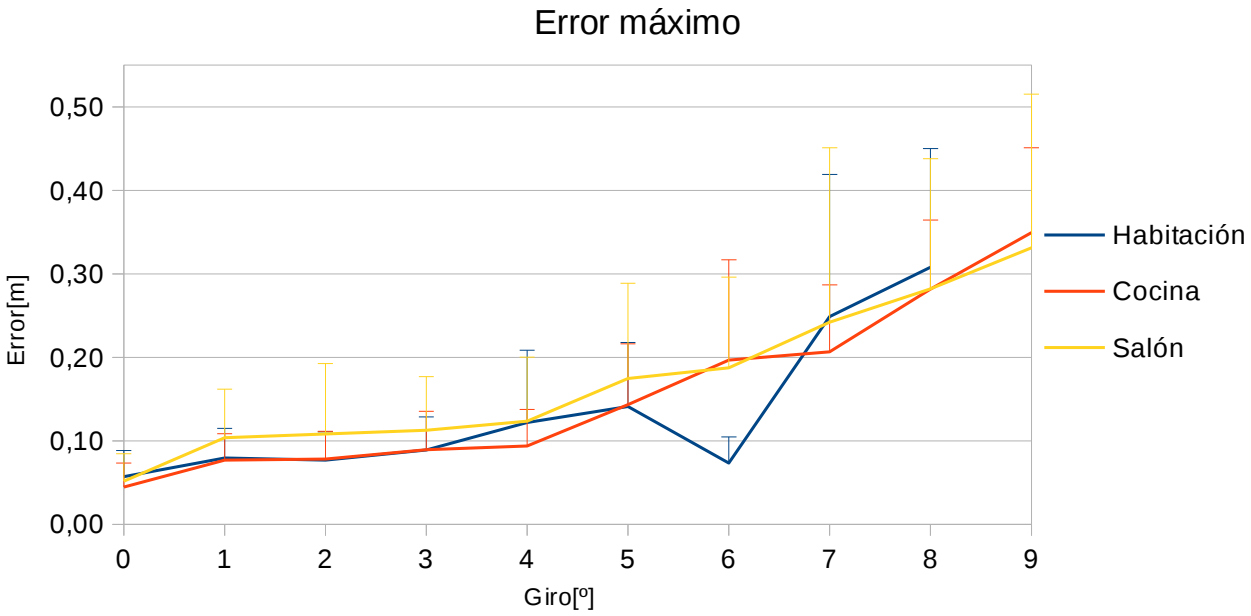


Con los cambios realizados no se consigue que las transformaciones obtenidas sean de mayor calidad, observándose divergencias a partir de los 200 mm al igual que en el caso anterior. La desviación típica sigue manteniendo valores altos.

Gráficas de movimiento rotación:







Para los giros, se logran atenuar las desviaciones típicas, por lo que hay mayor consenso en el cálculo de la transformación. Pero el error no es atenuado, con valores iniciales muy grandes, que no permiten un correcto registro entre nubes.

La metodología utilizada en este caso no ha redundado en mejores resultados de registros, el error medio mantiene valores pequeños con valores elevados para el error ponderado y máximo. Si bien, las desviaciones típicas indican menor divergencia de resultados, no es suficiente para conseguir resultados óptimos de registro. Es posible que los algoritmos no sean lo suficientemente restrictivos para comprobar que las correspondencias entre puntos sea lo suficientemente veraz como para que se atenúen los errores en una mayor medida.

Caso III

La disminución de puntos de interés no ha llegado a influir en la determinación de un buen solape. Se puede intuir que un descriptor muy corto no aporta suficiente información, puede ser una mejora que permita tener mejores emparejamientos. En los casos anteriores, los descriptores contenían una estructura de 33 datos, tomamos un descriptor que amplíe considerablemente esa estructura, intentando que los tiempos de procesamiento se mantengan estables.

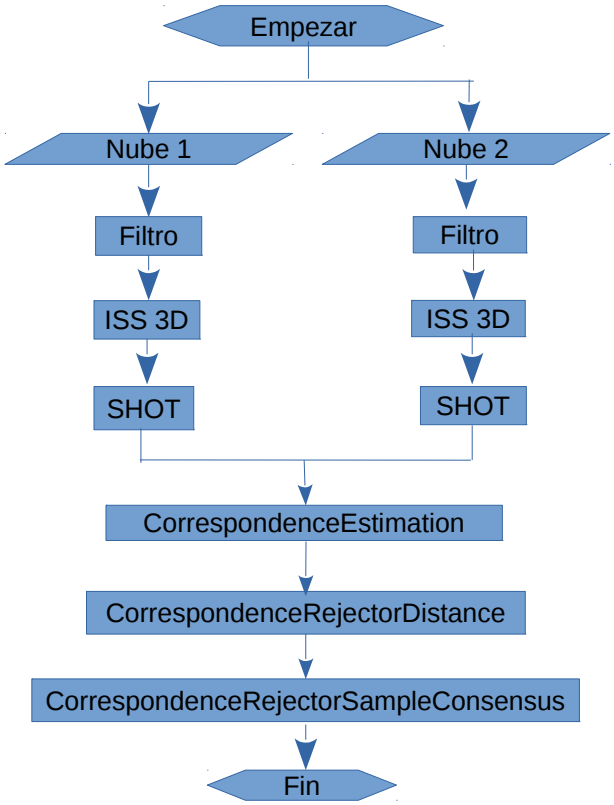
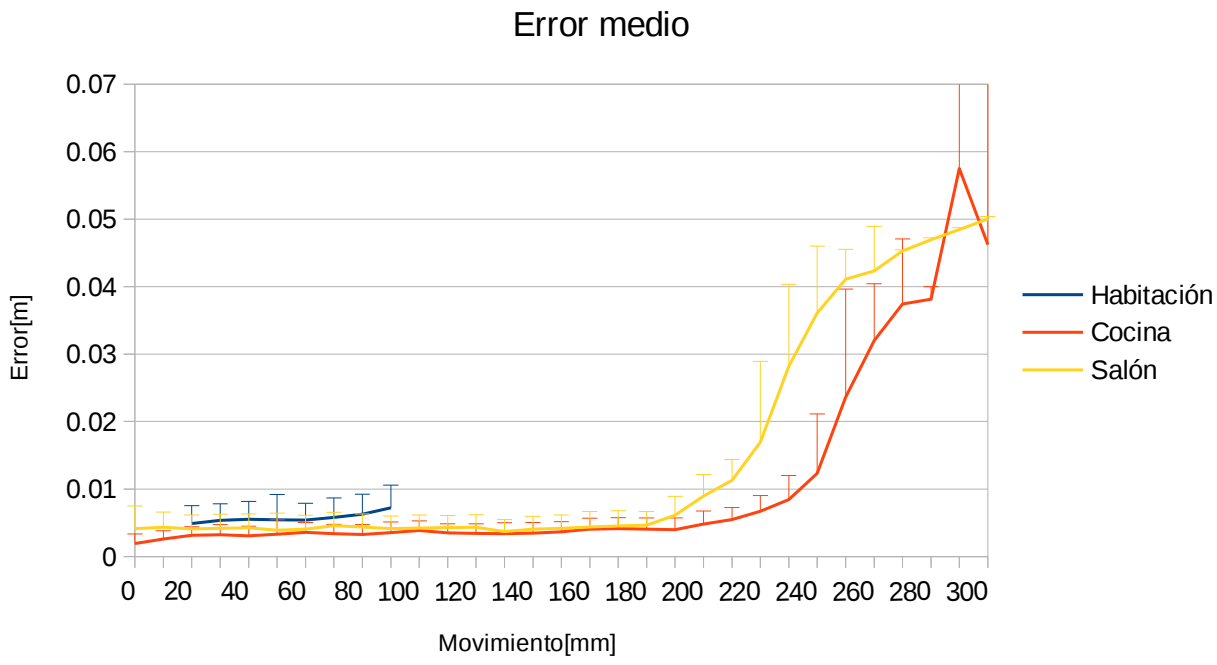
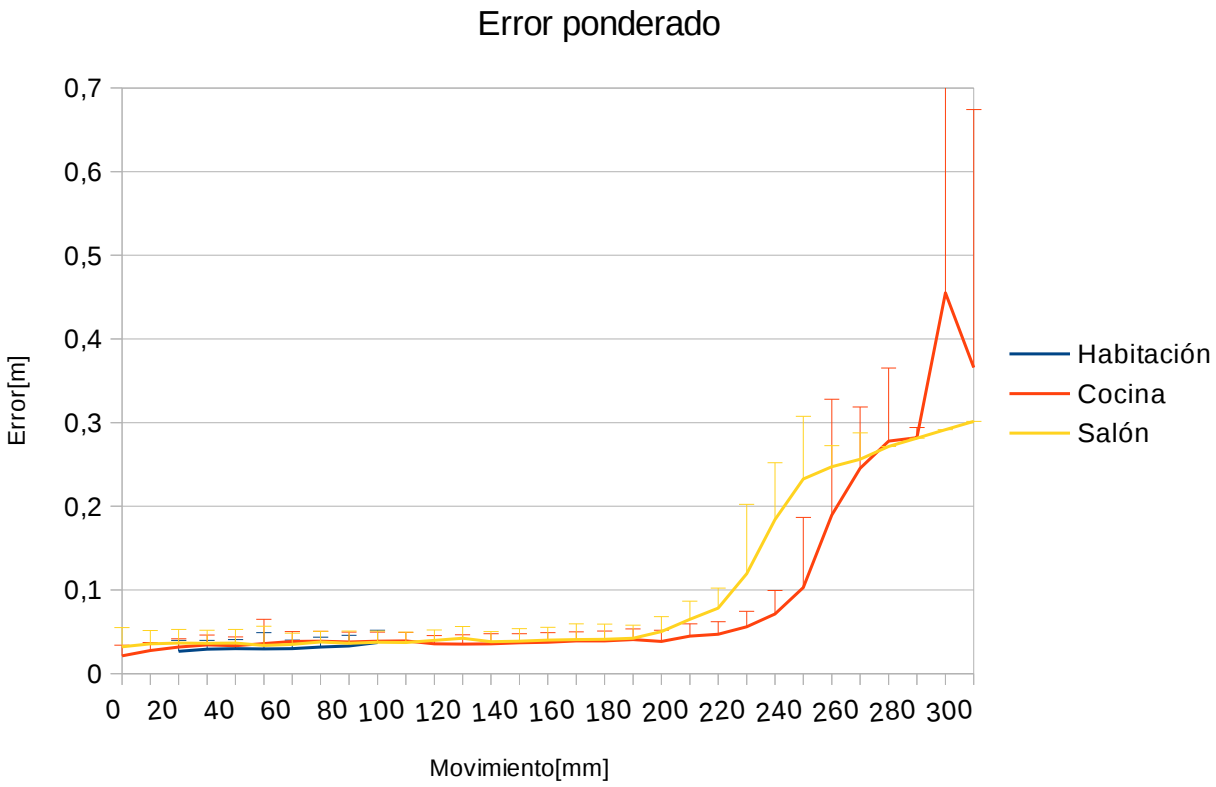


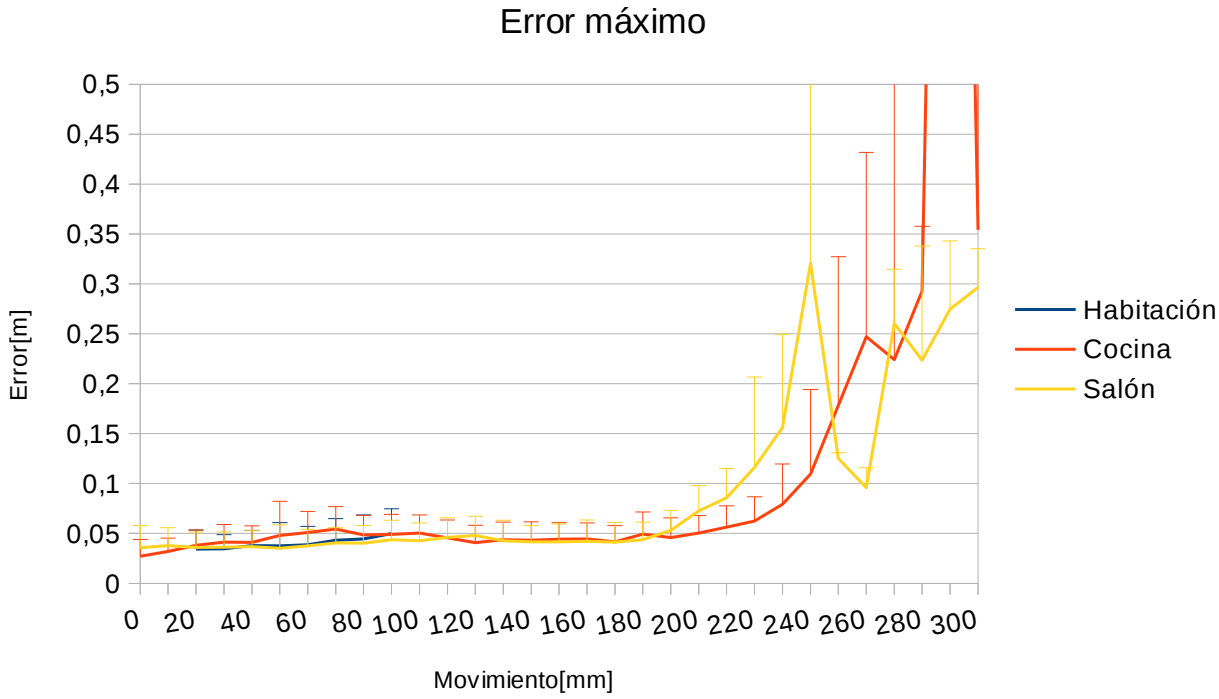
Figura 6.6: esquema de algoritmo para el caso III

Como las normales y los descriptores utilizan las clases pcl::NormalEstimationOMP y pcl::SHOTEstimationOMP respectivamente, los tiempos de ejecución hay que dividirlos entre el número de hilos.

Gráficas de movimiento rectilíneo;

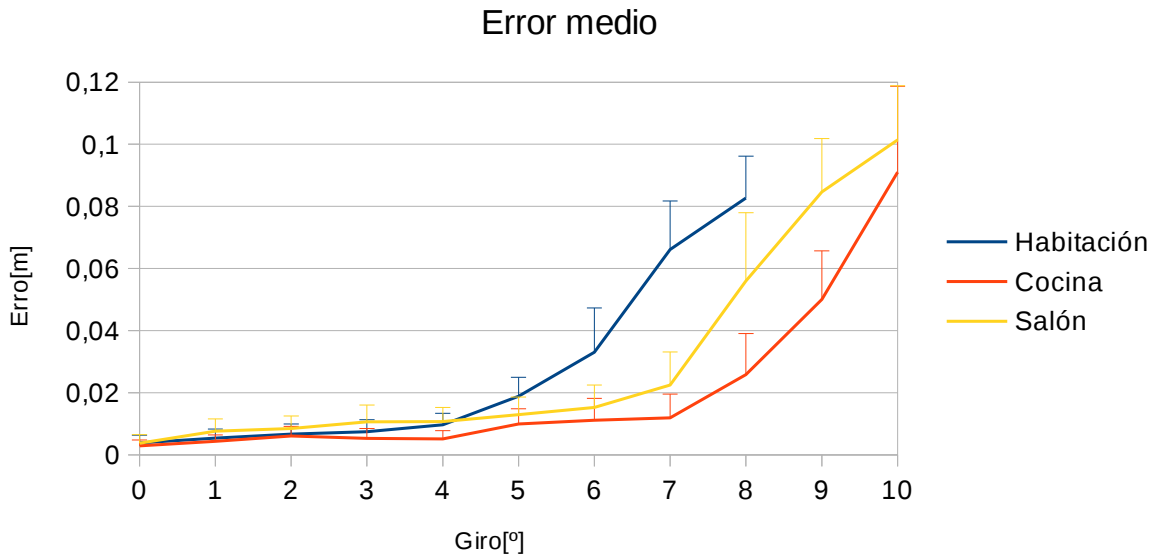


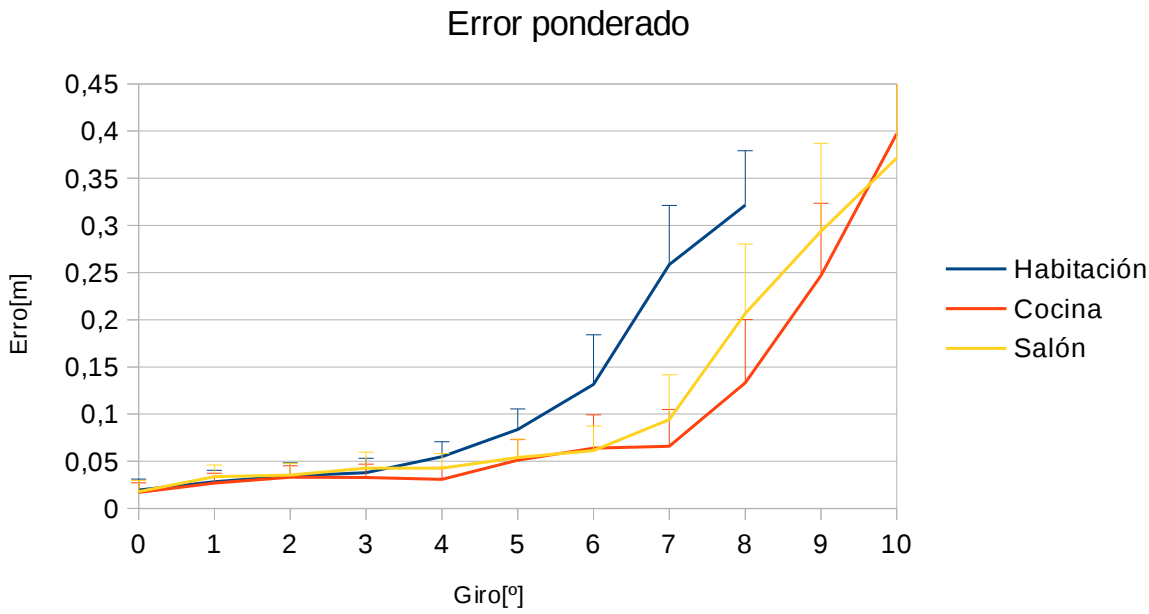

Error ponderado
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0
Error[m]
0 20 40 60 80 100 120 140 160 180 200 220 240 260 280 300
Movimiento[mm]
Habitación
Cocina
Salón

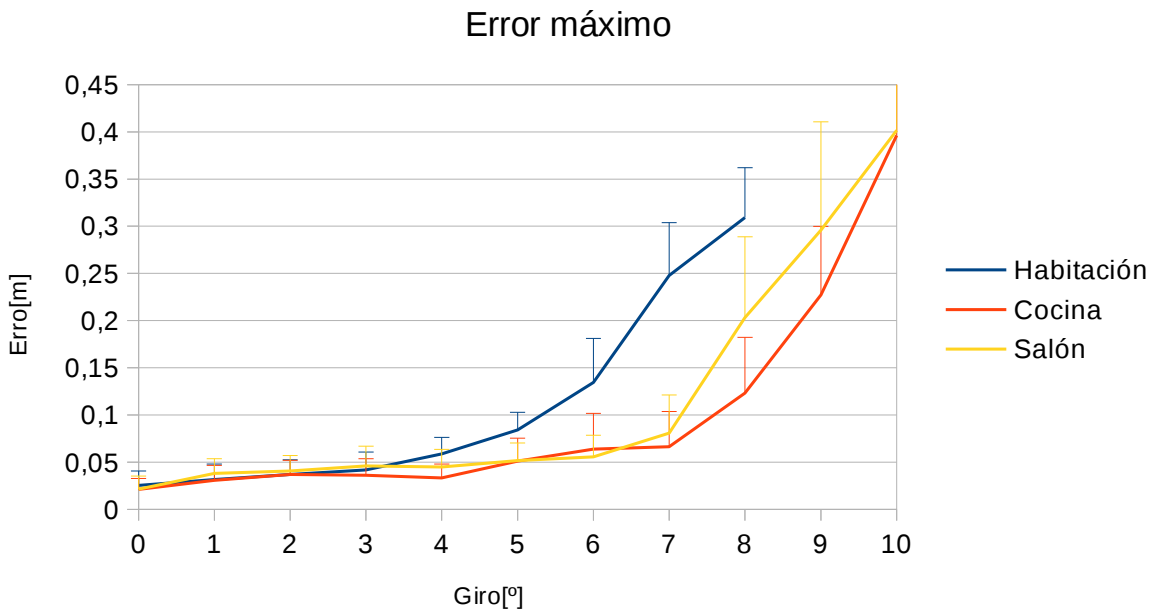

Error máximo
0,5
0,45
0,4
0,35
0,3
0,25
0,2
0,15
0,1
0,05
0
Error[m]
0 20 40 60 80 100 120 140 160 180 200 220 240 260 280 300
Movimiento[mm]
Habitación
Cocina
Salón

Los errores se atenúan en pequeña medida, la desviación típica se disminuye en gran medida, lo cual hace que el resultado tenga un error más predecible. Hay que destacar que ha mejorado mucho con respeto a los caso anteriores.

Gráficas de movimiento rotación;







Para los giros este caso tiene mayor estabilidad hasta los 5º, y aunque la precisión haya mejorado considerablemente aun es muy alto el error. Como se vio en el caso del movimiento rectilíneo, los cálculos tienen un mayor consenso para calcular la matriz de transformación, lo cual se refleja hasta los 5º donde el error ponderado y máximo aumenta muy poco con respecto al error inicial, producido por el registro de la muestra inicial.

De forma general este caso tiene mayor estabilidad para calcular el solape. La precisión ha mejorado con respecto a los casos anteriores, esto es debido al aumento de datos en los descriptores que permite una mejor comparación entre keypoints, para resolver el problema de correspondencias. Sin embargo, esto nos hace intuir un problema de resolución en las últimas etapas del registro, lo cual hace que las mejoras no sean lo suficientemente significativas para poder ser utilizado como método de registro.

Caso IV

Anteriormente, la modificación de los procesos ha supuesto una mejora en el solape de las nubes. Pero es necesario añadir procesos que disminuyan el error. Al enfoque propuesto por la librería PCL, se añade el proceso IPC (Iterative Closest Point). Con este método se termina de ajustar la nube mediante lo que se considera fuerza bruta, teniendo en cuenta toda la nube. Por tanto se espera tener un mejor ajuste que con los casos anteriores con los que compararemos sus diferencias con la matriz de transformación.

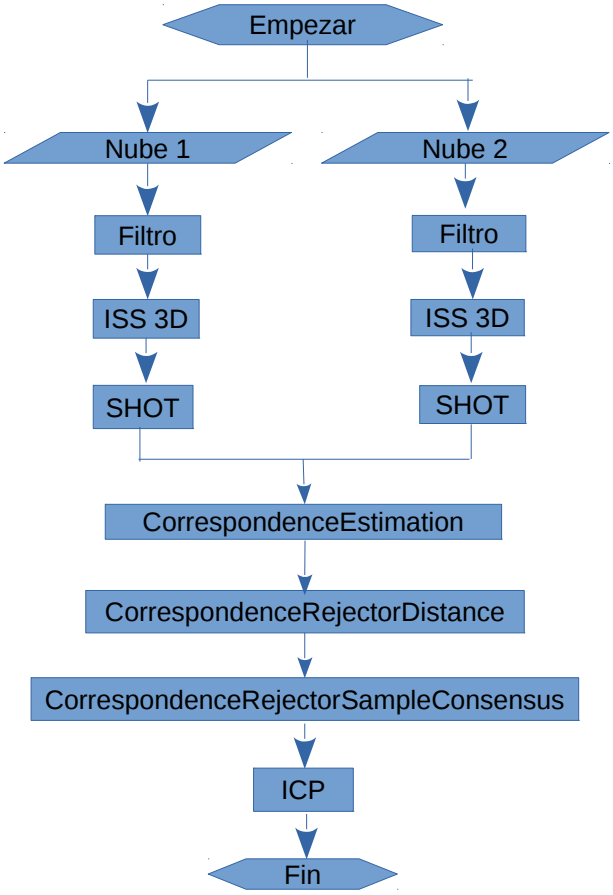
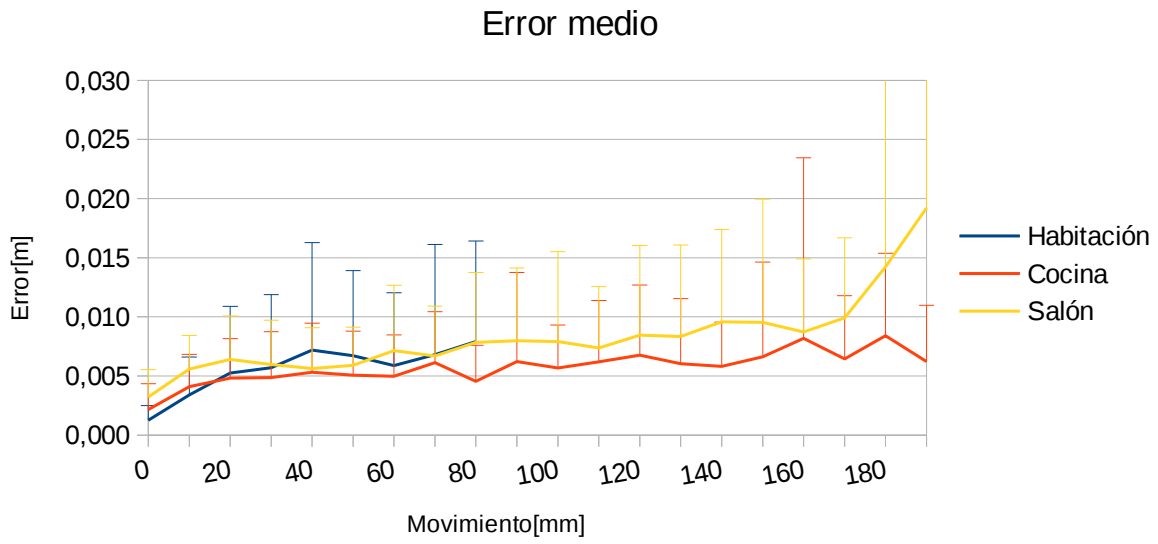
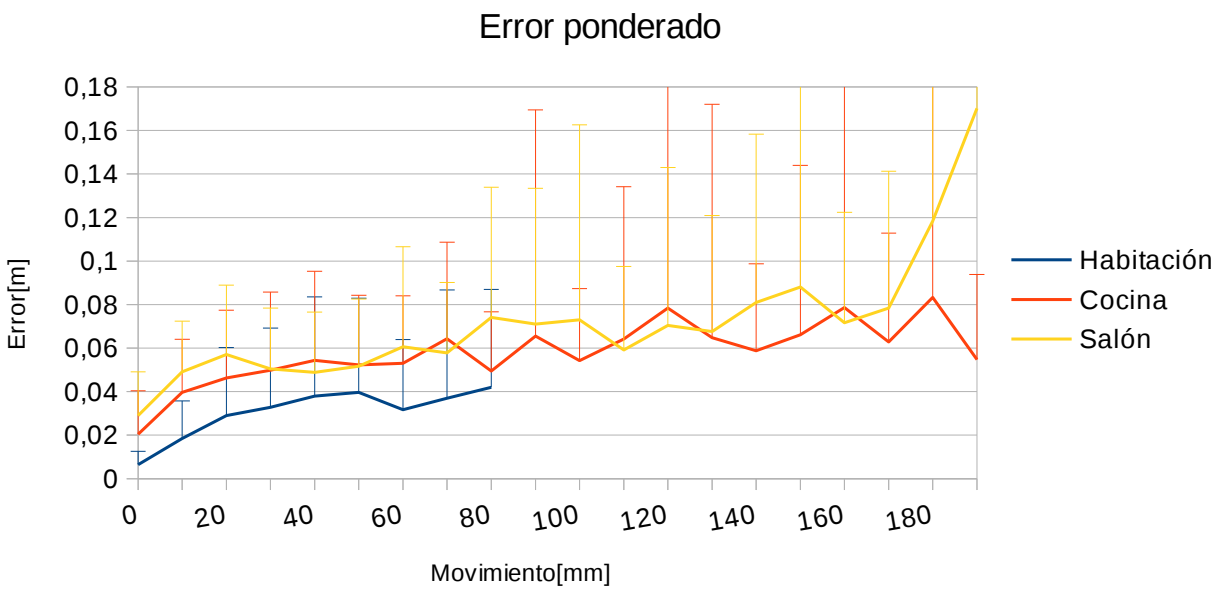


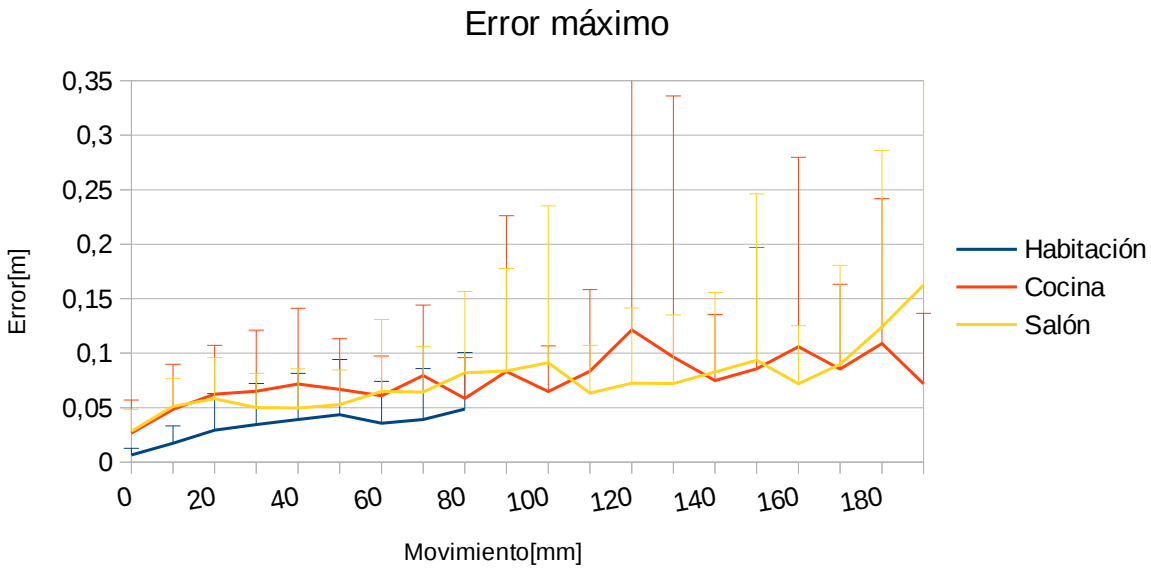
Figura 6.7: esquema de algoritmo para el caso IV

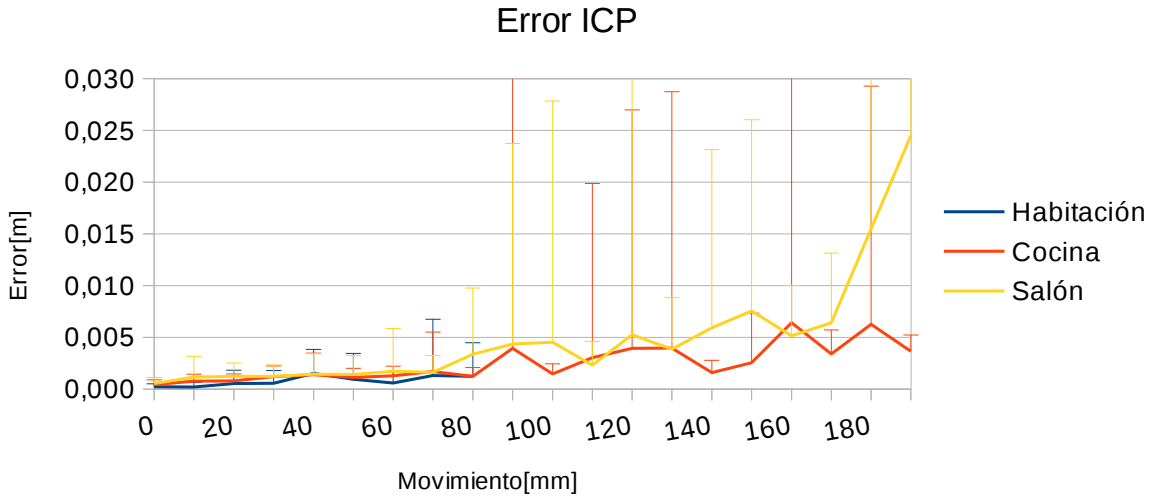
Como las normales y los descriptores utilizan la clases pcl::NormalEstimationOMP y pcl::SHOTEstimationOMP respectivamente, los tiempos de ejecución hay que dividirlos entre el número de hilos.

Gráficas de movimiento rectilíneo:



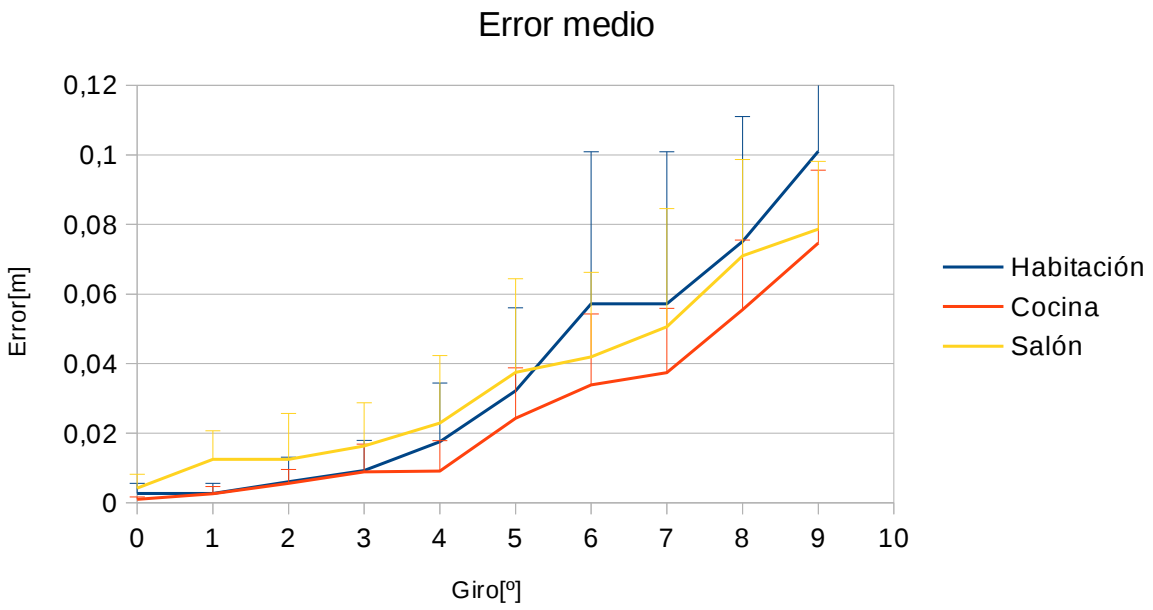

Error ponderado
0,18
0,16
0,14
0,12
0,1
0,08
0,06
0,04
0,02
0
Error[m]
0
20
40
60
80
100
120
140
160
180
Movimiento[mm]
Habitación
Cocina
Salón

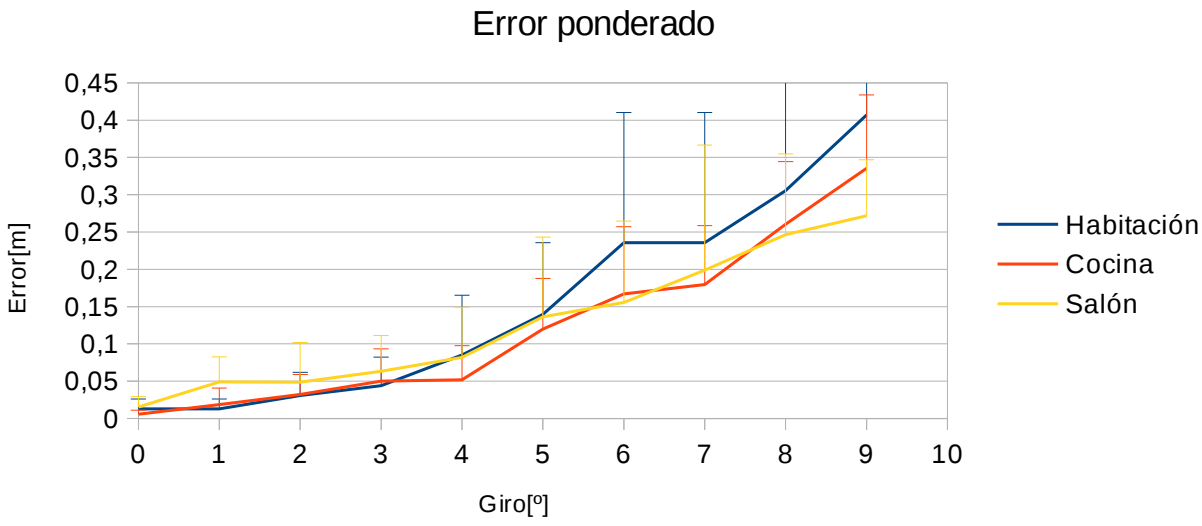

Error máximo
0,35
0,3
0,25
0,2
0,15
0,1
0,05
0
Error[m]
0
20
40
60
80
100
120
140
160
180
Movimiento[mm]
Habitación
Cocina
Salón

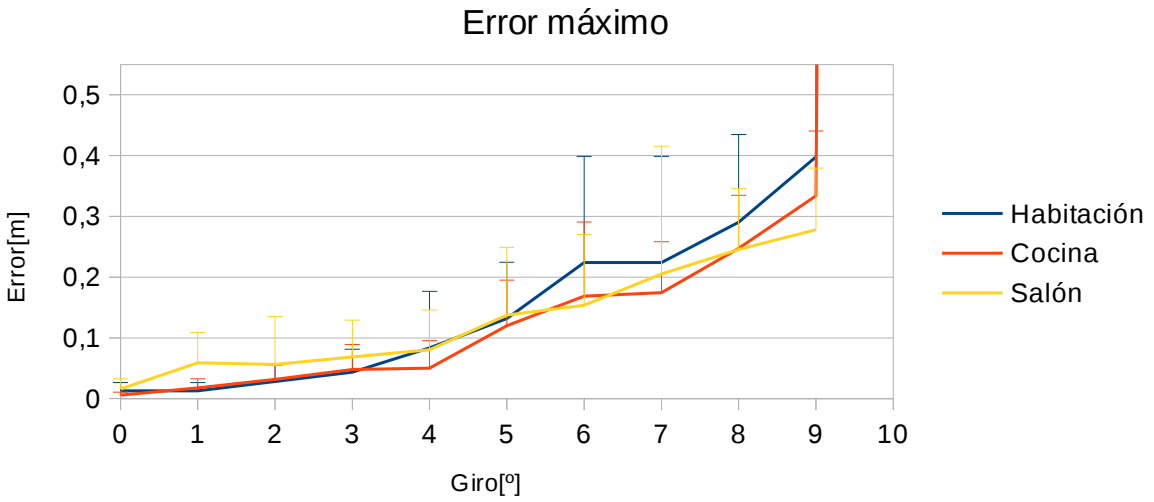

Error ICP
0,030
0,025
0,020
0,015
0,010
0,005
0,000
Error[m]
0
20
40
60
80
100
120
140
160
180
Movimiento[mm]
Habitación
Cocina
Salón

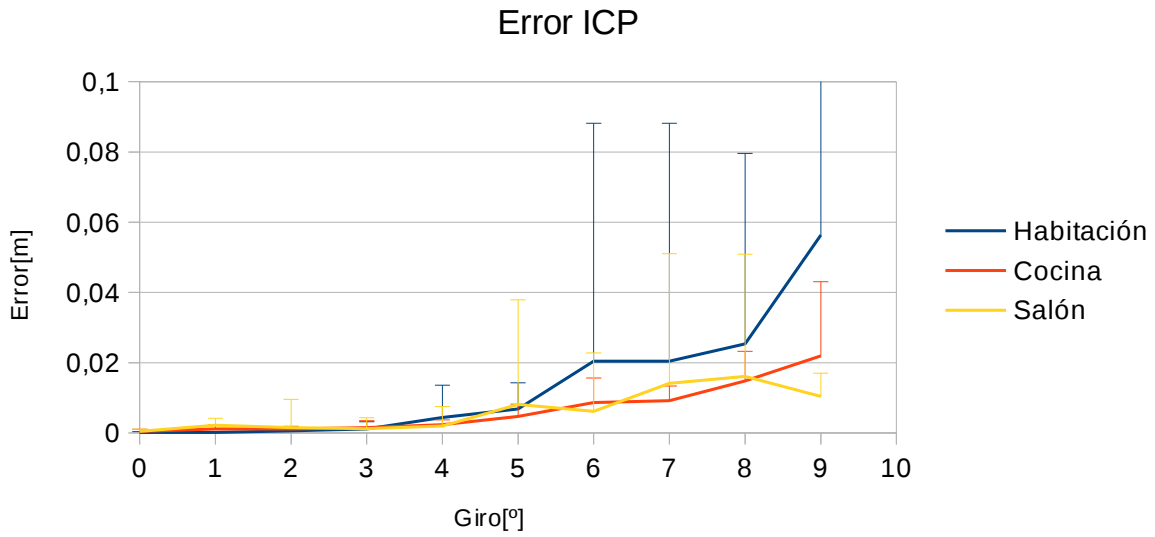
En este caso cuando se producen inestabilidades en los pasos previos no se puede conseguir una transformación mediante ICP. Es la causa por la que solo se calcula el error hasta los 200 mm, y en algunas escenas con reducción de la muestra tomada.

Gráficas de movimiento rotación;









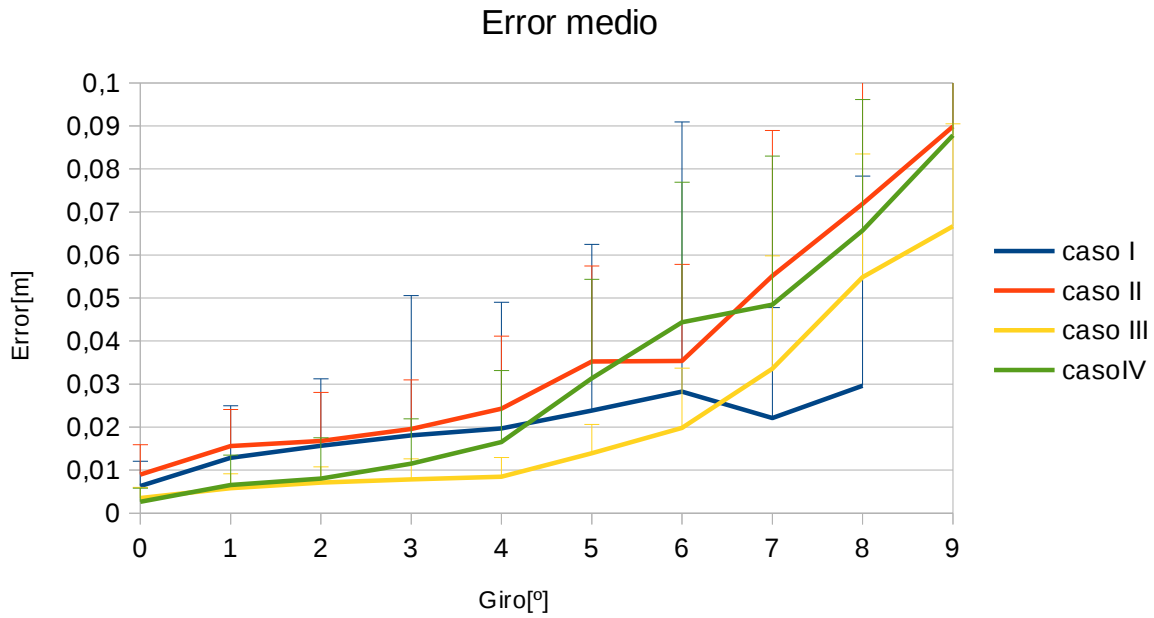
De las gráficas se deduce para giros superiores a 4º se entra en divergencia. Para valores inferiores se mantiene la estabilidad y la convergencia, pero los errores son muy elevados. Además, el error generado en este caso es mucho menor incluso que el error medio, por lo que determinamos que las correspondencias que calcula incluso en la últimas iteraciones, no son válidas.

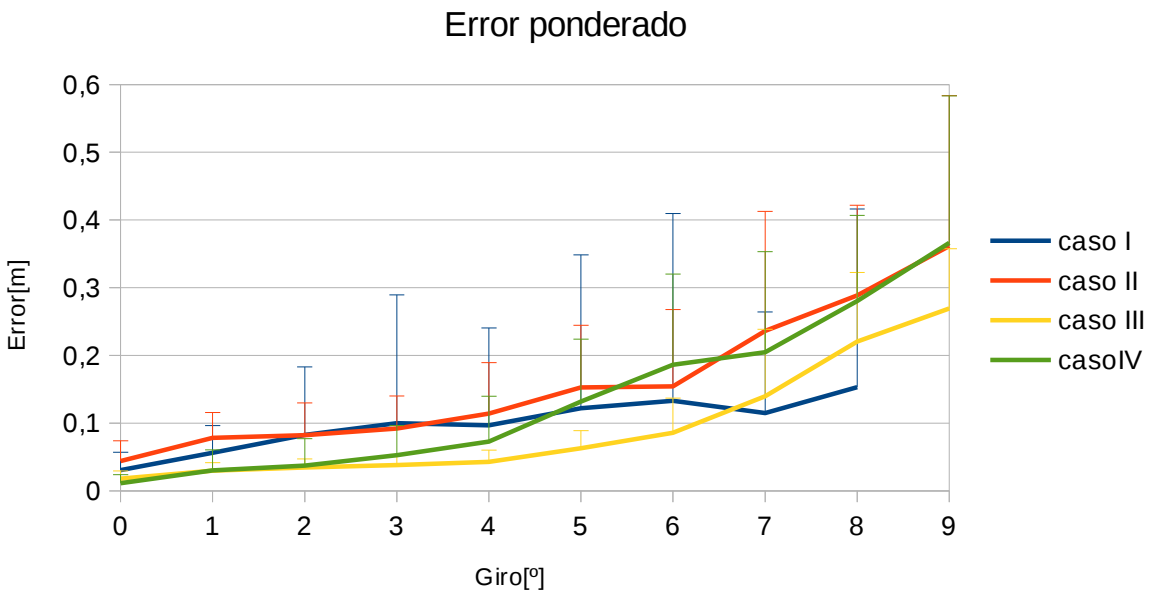
El algoritmo ICP no termina de solucionar el problema, se comporta bien para aproximaciones de unos pocos centímetros, pero entra en inestabilidades sin llegar a solucionar la transformación cuando no existe correspondencias posibles dentro del umbral definido como entrada del algoritmo. Por último, se observa como el parámetro de calidad del registro, debido a las correspondencias que realiza internamente ICP, obtiene valores muy pequeños lo cual nos indica que las correspondencias no son buenas debido a que se utilizan todos los puntos de la nube. Por tanto implementar en una última etapa del enfoque planteado por los desarrolladores de la librería no es un buena solución.

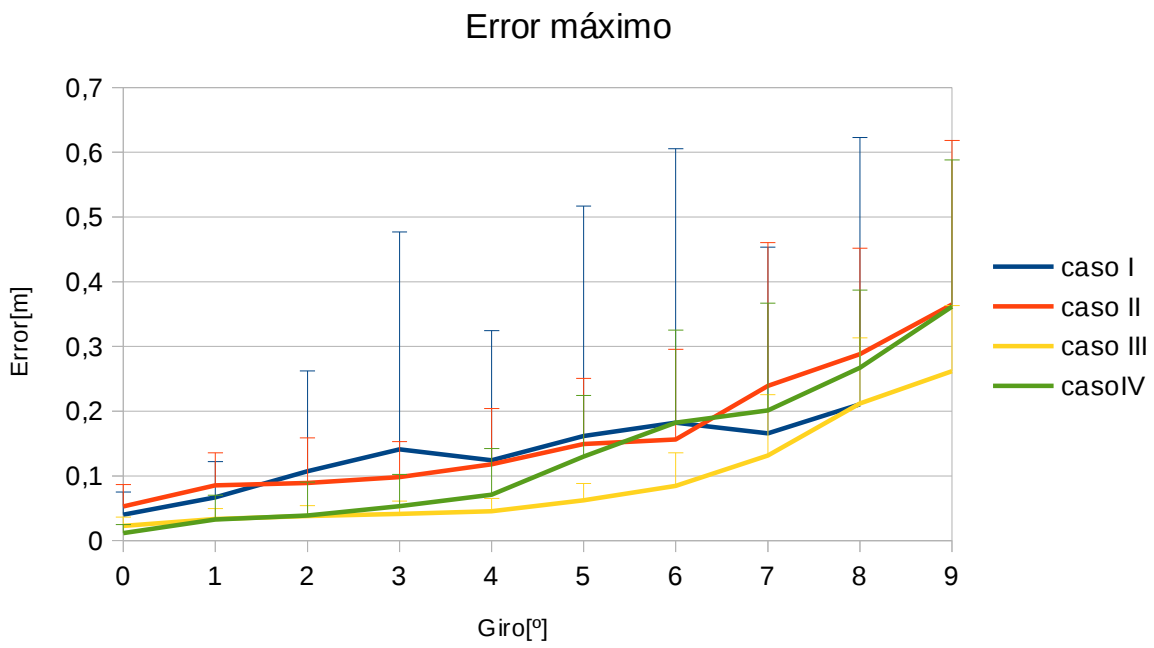
Resultados conjuntos

Para analizar los métodos es preciso comparar los diferentes casos de forma conjunta. Se tomarán los resultados de todas las escenas y se agruparan para calcular la media y su desviación típica, obteniendo así los resultados correspondientes a cada una de las distancias o giros. Ahora, en cada gráfica cada línea representa uno de los casos, y no como se mostraba anteriormente las diferentes escenas.

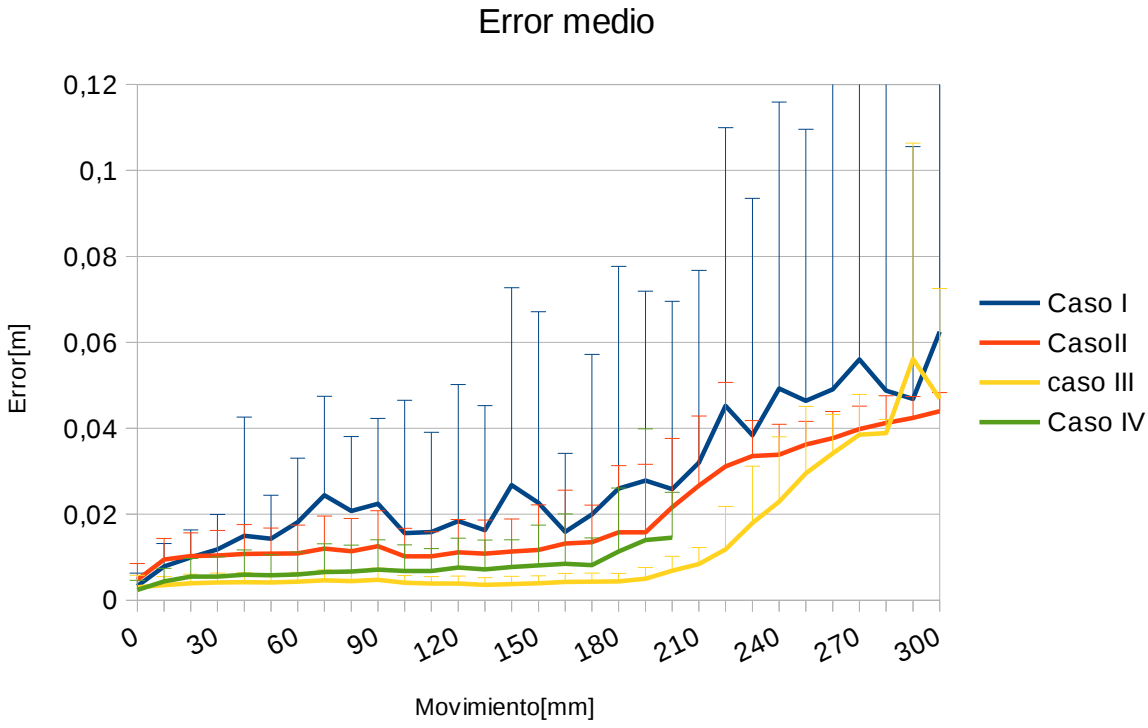
Gráficas de movimiento rotación.

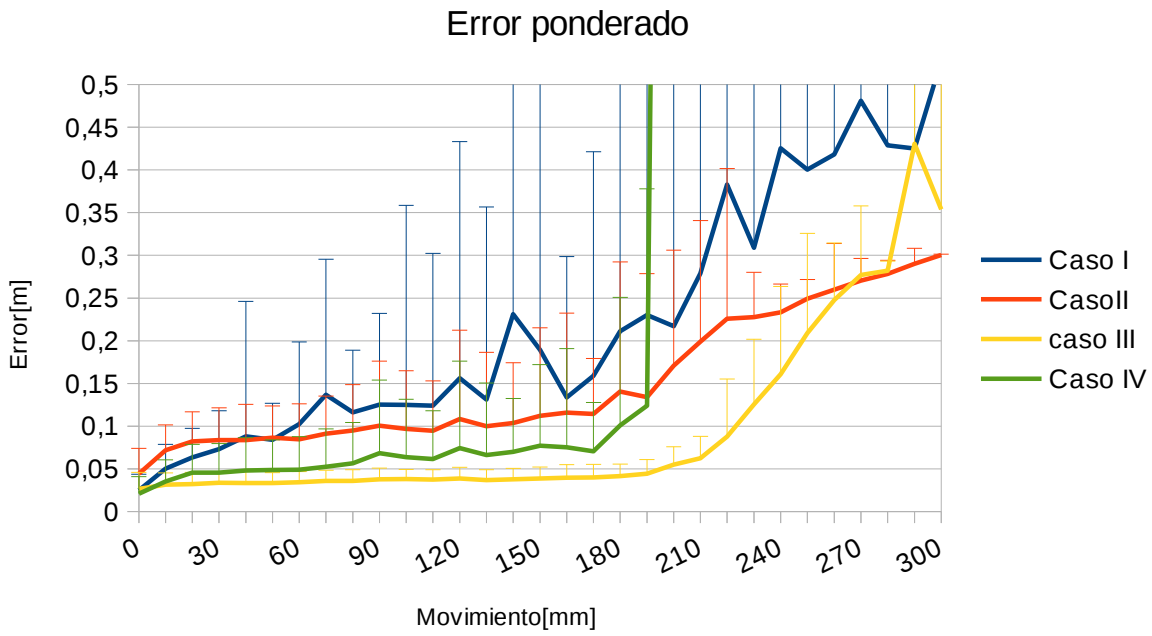


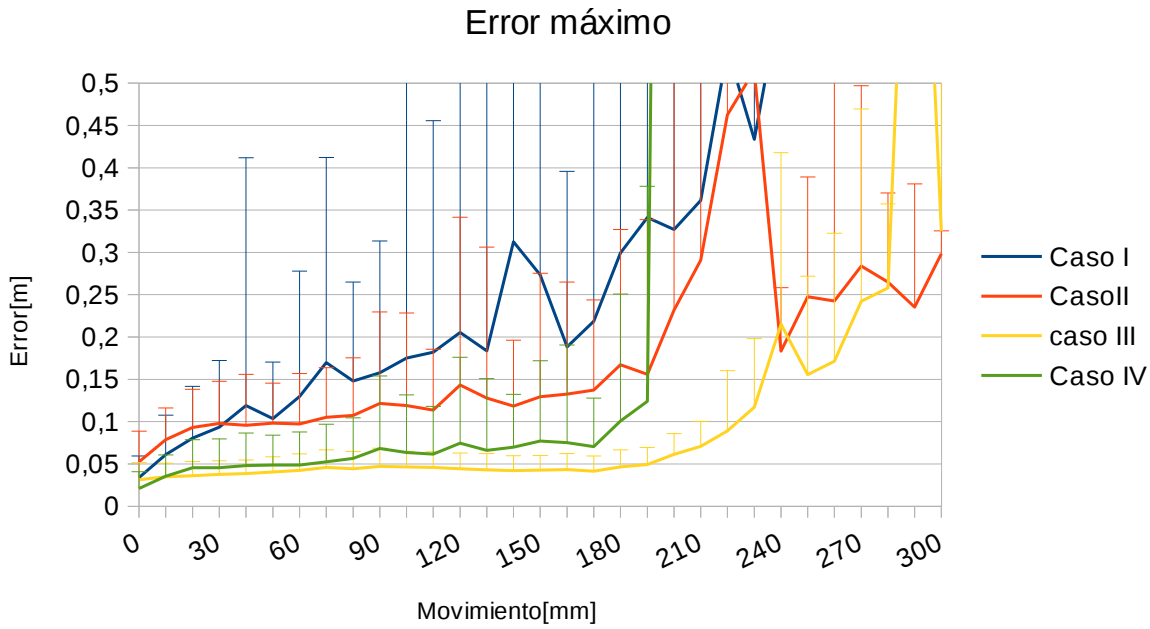




Gráficas de movimiento rectilíneo:







Tras estudiar cada caso de forma particular debemos de tener en cuenta los elementos comunes que tienen los diferentes casos. Por un lado, se observa que existe un error para desplazamientos nulos, lo cual indica que sin necesidad de experimentar un desplazamiento en los diferentes casos, los métodos utilizados no son capaces de alcanzar la solución correcta. Este fenómeno puede deberse a varios factores, como es el ruido contenido en la nube de puntos que no ha sido lo suficientemente filtrado, o su contenido es excesivo y ha penetrado en los algoritmos de las

diferentes etapas.

Existe en todos los casos estudiados un error entre el solape de nubes con el mismo sistema de referencia. Este error inicial indica la capacidad de convergencia del ajuste para los métodos utilizados en los diferentes casos.

Aunque podría ser trivial, por lo general el error aumenta cuanto más grandes son los desplazamientos producidos, y con ello también aumenta la desviación típica. Podría existir la posibilidad de que los errores fueran mayores debido a la toma de medidas, posibilidad que se descarta al existir divergencia en las desviaciones típicas. Esto se explica porque los métodos que ofrecen como válidos resultados no se ajustan, debido a las carencias de los métodos heurísticos para el cálculo de la matriz de transformación.

El movimiento lineal que estudiamos produce menos interferencia en las correspondencias por las distancias, por lo que la dificultad debería de ser menor para los algoritmos. Sin embargo, en todos los casos excepto en el caso III existan fuerte divergencias entre resultados, y los errores son muy elevados aunque se trate de un movimiento muy sencillo.

Si bien el giro estudiado es sencillo, cabe destacar que existe la posibilidad de que se entrelacen correspondencias debidas a la restricción de distancia. Se observa en el comportamiento del error en las gráficas, que un movimiento de 5º equivale a un movimiento lineal de más de 200 mm lo cual es una estimación cierta. Por lo tanto el comportamiento es similar para ambos movimientos.

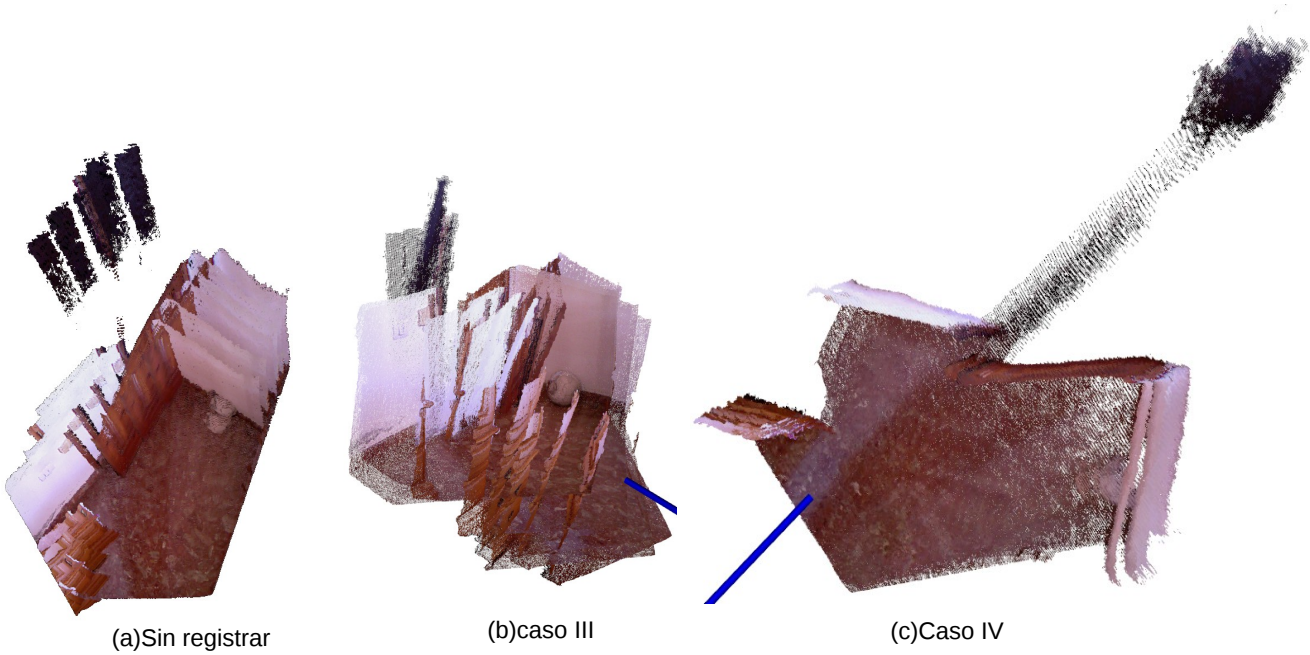
Los errores estudiados indican que existe una mayor sensibilidad del error, debido a incrementos de giros erróneos, que en mayor medida afectan al ajuste del marco de la imagen. Esto se puede comprobar analizando las diferencias entre el error medio y el error máximo. Por otro lado, el error ponderado muestra cómo afecta a toda la nube la desviación en el cálculo, y al ser el contorno una zona más amplia ponderarán más sobre este error.

En cuanto a la desviación típica que muestra la variabilidad de resultados de las muestras, se puede estimar como es de divergente el método aplicado en cada caso. La desviación típica no es correlación del error sino del movimiento producido ante la dificultad en cada caso para obtener la solución.

En el estudio de cada caso tenemos mejoras en cuanto a los resultados cuando se aporta una información de mayor calidad, esto quiere decir que es necesario seleccionar puntos de la nube que sean únicos (keypoints) y de estos extraer la mayor información posible (descriptores) que es lo que se ha conseguido en el caso III. El fracaso de la implementación del algoritmo ICP es debido a que no se tiene en cuenta un subconjunto de puntos de la nube sino está en su totalidad, por lo que es fácil que las correspondencias no sean veraces.

A modo de ejemplo, se han tomado una secuencia de nubes de puntos, trasladando el sensor a

una velocidad creciente, donde la distancia entre capturas va aumentando. Todas las nubes deben de ser transformadas para tener el mismo origen que la primera y se represente fielmente la realidad. En la figura 6.8(a), se observa cómo estarían las nubes sin transformar, y en la derecha después de aplicar los procesos de registro del caso III en la figura 6.8(b) y el caso IV en la figura 6.8(c) . Como se aprecia una de las nubes de puntos no se termina de solapar correctamente con el conjunto.



(a)Sin registrar (b)caso III (c)Caso IV

Figura 6.8: registro de conjunto de 9 nubes de puntos

En la figura 6.8 se puede observar visualmente cómo la precisión del conjunto de algoritmos utilizados en el caso IV no son lo suficientemente precisos para realizar un solape de nubes. Una de las principales causas es que los movimientos para los que se comporta bien son demasiado pequeños, y no podrían ajustarse ni utilizando métodos inerciales de predicción del movimiento. Por otro lado, en el caso III el resultado visual es mucho peor de lo que se espera, teniendo en cuenta los resultados del caso IV.

6.3 Registro mediantes múltiples etapas

En el apartado anterior se ha estudiado cómo se comportan las técnicas propuestas por PCL para el registro de nubes. Cabe destacar que en los tres primeros casos el registro solo consta de una etapa, en la cual se obtiene una matriz de transformación. Por otro lado, en el caso IV existen dos etapas donde existe una primera matriz de transformación, proporcionada por métodos similares a los casos anteriores y otra segunda matriz realizada mediantes el algoritmo de ICP, donde se refina el resultado final. Sin embargo, ninguno de estos métodos han sido capaces de ofrecer un resultado lo suficientemente eficaz para ser aplicado en la práctica.

En el registro de nubes de puntos que se propone, deben de aplicarse las restricciones que

describen un entorno interior, con escenas con formas esperadas. Esto quiere decir, que restricciones que haga el algoritmo robusto en la última etapa, es necesario que se cumplan las siguientes hipótesis:

1. La imagen es rígida.

2. Existen bordes lo suficientemente grandes y separados de forma moderada.

3. Existen un conjunto de puntos considerable que forman planos representativos de la imagen, junto con zonas de alta densidad de keypoints para restricciones débiles.

Al igual que en el método clásico necesitamos que la imagen sea rígida, para realizar transformaciones, de lo que se deduce que la realidad permanece inalterable entre diferentes tiempos de captura. Será necesaria esta hipótesis para que las siguientes hipótesis sean válidas en la transformación. Si bien no debemos descuidar la posibilidad de la capacidad que tiene el ruido para distorsionarla. Por lo tanto, no deben ocurrir eventos como movimientos debidos al viento, o desplazamientos de objetos o personas, sino debería existir una etapa previa que elimine esos elementos.

Si analizamos la segunda hipótesis, donde se requiere que la imagen tenga bordes, entendiendo los bordes como aristas de los objetos que están presentes en la imagen. No se entenderá como borde el marco de la imagen, que representa los límites de captura del sensor. Estos bordes deberán de ser amplios para poder ser identificados, ya que suponemos que están bien definidos y que son lo suficientemente grandes.

La tercera hipótesis, donde se hace referencia a la cantidad de puntos que pertenecen a planos representativos, nos indica que los planos representativos tienen una superficie considerable, ya que se supone que cuantificamos la superficie por la cantidad de puntos que la forman, ya que no existe un algoritmo que sea capaz de medir con precisión en unidad métrica.

Las segunda y tercera hipótesis son importantes en nuestro nuevo enfoque pues nos basaremos en un principio de registro de formas geométricas, donde necesitamos analizar las imágenes en busca de formas geométricas las cuales están presentes en dos imágenes que se pueden solapar y registrando ambas imágenes en función de los objeto presentes. Los puntos de interés pueden estar afectados por el ruido, sino cuando son calculados, si pueden afectar los vecinos, variando los datos de sus descriptores de un mismo punto de la realidad en diferentes nubes de puntos.

Sin embargo, si buscamos formas geométricas, el análisis que realizamos de la imagen es a nivel macroscópico, si bien es posible encontrar varias formas geométricas con las mismas dimensiones, a estas se les podrá aplicar el principio de cercanía para su correspondencia, en donde se prevé el mínimo movimiento.

En algoritmo previo no está desarrollado completamente para realizar un registro completo, sino que pretende ser una etapa de acercamiento o aproximación. Con esto se consigue una mayor cercanía de las correspondencias entre keypoints. Se realizan dos acercamientos, uno mediante los bordes en búsqueda de las rectas más representativas, y un segundo acercamiento en el que se busca el del alineamiento de planos.

El primer acercamiento, mediante el acercamiento del emparejamientos de rectas, es necesario realizar el paso previo de la búsqueda de las rectas, para ello es preciso extraer de la nube sus bordes. El mecanismo para obtener los bordes ha sido desarrollado en los métodos de obtención de keypoints, si bien los bordes extraídos son un conjunto de puntos considerados keypoint, pero que serán tratados previamente para utilizarlos como borde con otros parámetros. Para extraer los bordes en PCL existen funciones de imágenes de rango, mediante las cuales se extraen los diferentes tipos de bordes. En la siguiente imagen se muestran los puntos extraídos como bordes, estando estos resaltados.

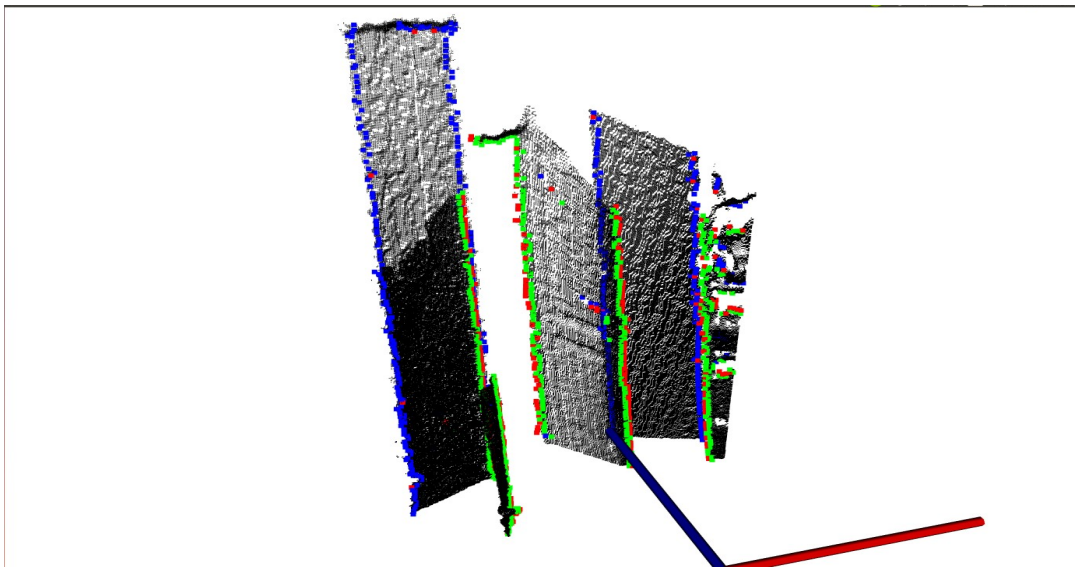
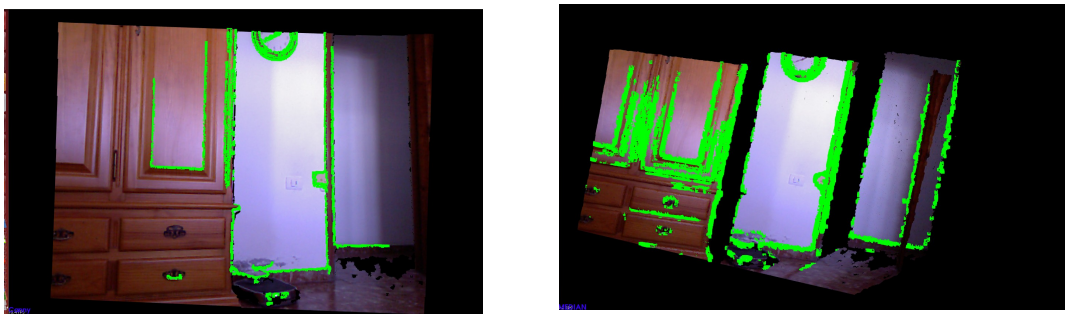


Figura 6.9: nube de puntos con los bordes resaltados en color azul, verde y rojo.

También existen métodos de obtención de bordes que forman líneas, estos son heredados del procesamiento de imagen 2D, donde se realiza una convolución para extraer los píxeles que forman bordes. Si se trabaja con nubes de puntos ordenadas, como las que entrega el sensor Kinect, se puede convertir en imagen digital 2D, con lo cual cuando se cambia de formato debe establecerse la correspondencia que existe entre píxeles y puntos. Para luego procesar los puntos de la nube donde se obtienen las líneas.

Para la extracción de bordes por cambios de intensidad del color, es posible aplicar algoritmos desarrollados por la librería OpenCV, como es el detector de Canny expuesto en las figura(a), o por

poner otro ejemplo aplicación de una convolución laplaciana, en la figura(b) con umbrales de discriminación.



(a) (b)

Figura 6.10: extracción de bordes mediante Canny (a) y laplaciana (b).

6.3.1 Registro de “líneas rectas”

Las ecuaciones que representa formas geométricas tienen la particularidad de que con pocos datos representamos un amplio espectro de imagen. Sin embargo, una misma figura geométrica tiene las mismas dimensiones, pero los valores de los diferentes parámetros son diferentes debido al punto de vista desde donde se ha realizado la imagen. Por tanto, es importante tener en cuenta aquellos parámetros que son invariantes al punto de vista y los parámetros que varían, pues el primero servirá para realizar las correspondencias, y los segundos servirán para realizar las transformaciones necesarias.

Las rectas, no son como tal figuras geométricas, pero se representan matemáticamente de forma sencilla, con una ecuación de primer grado con tres variables que corresponde a la dirección de la misma. Estas no están acotadas, estableciendo unas dimensiones, por lo que tenemos que introducir el concepto de línea rectas, donde se introduce la particularidad de tener una recta con un principio y un final. Con este concepto se indica las dimensiones de las líneas rectas, lo cual es importante, para realizar correspondencias entre líneas rectas.

Además para realizar las correspondencias, se aplican umbrales que eviten errores. Los umbrales solo pueden aplicarse sobre dos características esenciales de las líneas rectas, y es la establecida por el acotamientos de las rectas, pues no es factibles extraer dos líneas con una precisión absoluta que permita una correspondencia inevitable, por lo que dos líneas rectas en diferentes imágenes tendrán correspondencias si sus longitudes son similares, este umbral lo denominaremos $L_{1\text{-}2}$, dicho umbral expresa la diferencia máxima entre dos líneas rectas en las que existe correspondencia.

Debido a los desplazamientos del sensor existirá un desplazamiento de las rectas, pero al no

existir desplazamientos muy grandes, es conveniente aplicar un umbral sobre la distancias que se desplazaría la nube si existiera una correspondencias, este umbral se define como $d_{1\text{-}2}$, en la cual se establece la distancias entre las líneas rectas. Para calcular la distancia entre dos rectas que no se cortan, suponemos las rectas *r* y *s*, y donde la distancia mínima d(*r*,*s*) será:

$$d(r,s)=\frac{|[\vec{v}_r,\vec{v}_s,\overrightarrow{p_r p_s}]|}{|\vec{v}_r\times\vec{v}_s|} \tag{6.6}$$

donde $\bar{v}_r$ y $\bar{v}_s$ son los vectores directores de las correspondientes rectas, y $p_r p_s$ es el vector que une cualquier punto de la recta *r* con otro cualquiera de la recta *s*, obteniendo mediante la ecuación 6.6 la distancia mínima entre dos rectas. Sin embargo, en el problema que se plantea sobre correspondencias de líneas rectas, donde existe una acotación del elemento, existe la posibilidad que la distancia mínima no corresponde entre dos puntos de la acotación, Por lo tanto el parámetro de la distancia mínima, lo desestimamos aunque podría ser válido si se somete a verificaciones sobre el acotamiento.

Sin embargo debe existir una distancia entre las rectas que debe ser umbralizada, el más sencillo es calcular el punto bisectriz, o centro de gravedad de la línea, este concepto es más sencillo porque no es necesario trabajar directamente con las líneas, sino que tomando los conjuntos de puntos que conforman una línea se puede aplicar la ecuación de centros de gravedad para sistemas discretos:

$$r_{cdg}=\frac{\sum_{i=0}^{n} m_i r_i}{\sum_{i=0}^{n} m_i} \tag{6.7}$$

donde r_{cdg} es el vector que une el centro de gravedad con el sistema de referencia, r_i son los vectores debidos a cada punto, y m_i es la ponderación de cada punto del sistema. Este método es más sencillo, pero puede ser que las distancias sean mayores que calculando la bisectriz, ya que influye cómo estén distribuidos los puntos sobre la línea, aunque se interpreta que la distribución debe ser semejante.

Por último, es necesario introducir un umbral, que permita distinguir la direcciones de las rectas, pues aunque los umbrales anteriores sean admitidos, existe la posibilidad de tener el caso más drásticos y es que las rectas sean ortogonales. Por eso es importante que ambas rectas tengan vectores directores similares, acotando un ángulo máximo como umbral para que entre ambas líneas exista correspondencias, el cálculo es de la siguiente manera:

$$\alpha_{umbral}=u_r\times u_s \tag{6.8}$$

siendo u_r y u_s los vectores directores unitarios de ambas rectas, y α_{umbral} es el producto vectorial de ambos, lo cual corresponde al coseno del ángulo entre ambos vectores, por tanto tendrá que ser lo

más cercano a la unidad.

6.3.2 Transformación por “líneas rectas”

Una vez se saben qué líneas de la imagen fuente corresponde a las de la imagen objetivo, se puede iniciar la tarea de calcular la matriz de transformación que permita tener un punto de la realidad un mismo sistema de referencias para ambas imágenes. Sin embargo el proceso que se llevará a cabo solo servirá para realizar un acercamiento que evite tener umbrales más relajados en las tareas de correspondencias clásicas. Por lo que solo se realizarán movimientos de traslación, uniendo los centros de gravedad de las líneas en las que existan correspondencias.

No se ha intentado realizar una transformación exacta debida a las líneas, porque las líneas más fáciles de obtener son debidas a los contornos, por lo que muchas de las líneas que pertenecen a unos contornos de un objeto no son exactamente las que corresponden a la realidad. Aunque en el caso de convoluciones en 2D es posible que sí que existan correspondencias reales ya que no solo se hayan contornos.

6.3.3 Registro de “superficies planas”.

En una imagen en 3D uno de las figuras geométricas que más predominan son los planos, ya que incluso otras figuras geométricas están formadas por el conjunto de varios planos acotados. Sin embargo, al igual que ocurre en el caso de las rectas, los planos son superficies no acotadas. Los planos al igual que las rectas están representadas, por ecuaciones de primer grado donde se define el vector normal al plano y una constante que desplaza al plano del origen. Siendo la siguiente ecuación la representación matemática del plano:

$$a \cdot x + b \cdot y + c \cdot z + d = 0 \tag{6.9}$$

donde a,b,c son las componentes del vector normal al plano. Esta ecuación es sencilla de manejar en vez de un amplio conjunto de puntos, en los que solo se podría extraer características de color y no geométricas, por lo que se evita manejar un gran número de datos, que aportan poca información. Sin embargo, las superficies de las que se extraen los puntos están acotadas, por lo que es necesario introducir el concepto de superficie plana, figura geométrica acotada, sobre las que se necesitan más parámetros para poder trabajar con estas superficies.

En el problema que se puede establecer en el registro de planos, donde tenemos una imagen fuente sobre la que necesitamos registrar otra imagen objetivo, es necesario que ambas imágenes tengan al menos una superficie real en común para solucionar el problema. Este registro se podría completar con una superficie, si se cumple la condición de tener la acotación real de la superficie completa, algo que no siempre es posible debido a las sombras de otros objetos en la toma de la

imagen. Para esto sería necesario implementar un algoritmo que tuviera en cuenta el contorno.

El proceso de obtención del contorno de una superficie plana no regular genera datos, que en algunos casos puede ser de un gran volumen, por lo que se generaría un problema similar al que ya existía, esto hace necesario encontrar un parámetro simple mediante el cual se obtenga información suficiente para realizar una aproximación entre ambas nubes en las cuales los parámetros de umbralización sean lo más restrictivo posible. Para tener la información con a las características de la superficie es necesario calcular la superficie sobre la que se desarrolla el plano, y el centro de gravedad sobre donde se ubica. Esta información simplifica mucho el contorno de la nube de puntos necesaria para definir el contorno.

Para calcular el centro de gravedad de la superficie es necesario aplicar una integral definida sobre la acotación de la superficie, lo cual es un problema que hay que resolverlo discretizando en computación, por lo que existiendo una nube de puntos, se puede resolver el problema computacional aplicando la fórmula aplicada para el caso de las líneas rectas acotadas en la ecuación 6.7, con el conjunto de puntos que forma parte del planos. En el caso de la superficie que pertenece al plano se genera el mismo problema, donde no es necesario los elementos de acotación, sin embargo se puede realizar una aproximación que puede generar un error, y es realizar la equivalencia por número de puntos pertenecientes al conjunto de la superficie.

Para la realización del registro es necesario el paso previo de las correspondencias entre planos, para los cuales hay que asentar las bases para aplicar los umbrales a los que se somete las comparaciones entre las diferentes superficies planas. Es necesario comparar la información de la que disponemos, de ambas nubes. En adelante se enumeran y se explica como funciona cada parámetro a umbralizar en orden de importancia:

1. Superficie: se estima como el número de puntos que pertenecen al plano y son de la imagen, y deben tener valores similares en ambos casos, por lo que su diferencia debe ser lo más pequeña posible.

2. Paralelismo: la ecuación del plano contiene el vector ortogonal al plano, por lo que si se realiza un producto ortogonal entre dos vectores de diferentes planos deben ser lo más cercanos a la unidad para que exista una relación por paralelismo.

3. Posición: para que corresponden dos planos es necesario que exista proximidad entre ellas de los centros de gravedad.

Una vez obtenidas las correspondencias entre planos, se realizan las tareas de transformación de la nube, en la transformación que se aplica no se producen movimientos rotatorios de la nube fuente para registrarla sobre la objetivo, si bien podría aplicarse no existen algoritmos para hacerlos, por lo que solo será una tarea de aproximación.

Para realizar la tarea de aproximación, se tendrá en cuenta desplazamiento ortogonales entre planos desplazando la distancia mínima entre el punto centro de gravedad de la superficie objetivo y la superficie fuente.

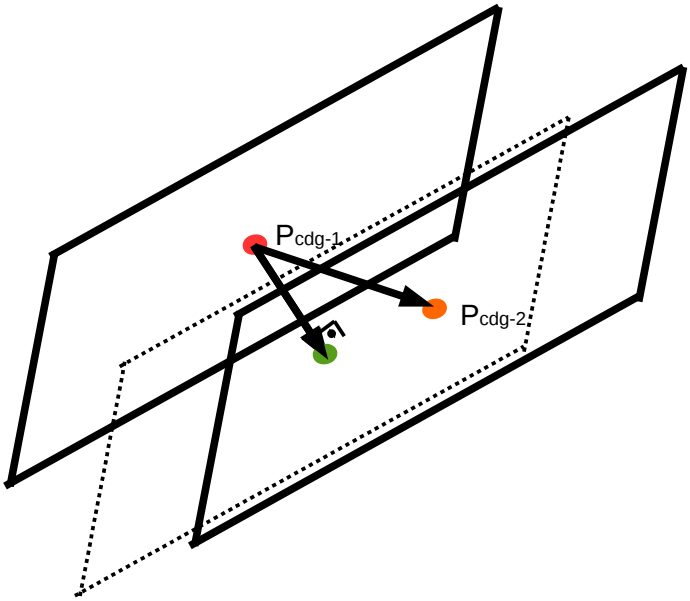


Figura 6.11: ejemplo de desplazamiento por correspondencia de planos

En esta transformación se calcula el vector de desplazamiento de la nube fuente, por tanto cuando existen correspondencias entre varios planos, es bastante probable que se machaquen los datos de cada sentido de desplazamiento, por lo que es importante que solo se guarden los datos en los que exista el máximo desplazamiento. También es posible aplicar restricciones por la aplicación relajada de los umbrales impuestos en las correspondencias de los planos.

6.3.4 Registro de Keypoints.

Para la etapa final de refinado hemos comprobado que es válido realizarla mediante el método de keypoint. En los métodos anteriores se ha comprobado que cuando los desplazamientos del registro son cortos el error es menor y consigue ser aceptable. Sin embargo, como no confiamos en ningún método de obtención de keypoint en concreto, se aplicarán técnicas de multihilo para procesar simultáneamente en paralelo las nubes de puntos y obtener varias nubes de puntos de interés.

6.3.5 Estudio del método multietapas

En los anteriores apartados, se explicó los principios que se utilizan para un registro aproximado de las nubes de puntos, mediante la extracción y correspondencias de líneas y planos. Todo esto sirve como paso inicial de aproximación, que estará a su vez dividido en dos etapas de aproximación, un acercamiento mediante las líneas, con el cálculo los términos referidos a la traslación de los puntos. El segundo paso, se extraerán los planos que servirá para el cálculo de los

términos de los giros en la matriz de transformación.

En la segunda etapa se determinarán los keypoints mediante diferentes métodos, como Harris, SUSAN, ISS_3D, AGAST. Para determinar las correspondencia de los pares de nubes se obviaron pasos intermedios aplicado en los métodos anteriores, como el estudio de características o *feactures*. Para obtener el registro definitivo y refinado de los pares de nubes se realizará mediante el algoritmo iterativo ICP, sobre este algoritmo se realimentará con los diferentes nubes obtenidas con los diferentes métodos de obtención de puntos de interés, con orden ascendente de cantidad de puntos de la nube. En cada iteración sobre el algoritmo de ICP se calculará la transformación de los los datos que realimentan en la siguiente iteración.

En el siguiente esquema se representa como funcionará el método que a partir de ahora lo denominaremos como método V para abreviar en los gráficos desarrollados en este apartado. Si bien no se puede representar de manera precisa la utilización de ejecución multihilo, que permita la programación paralela.

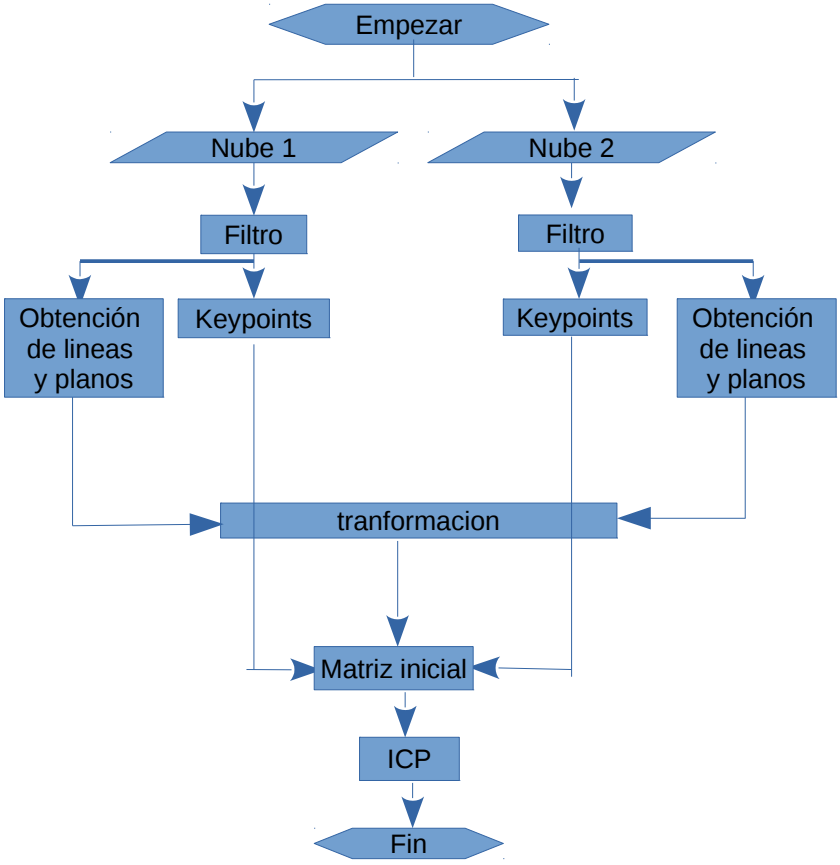
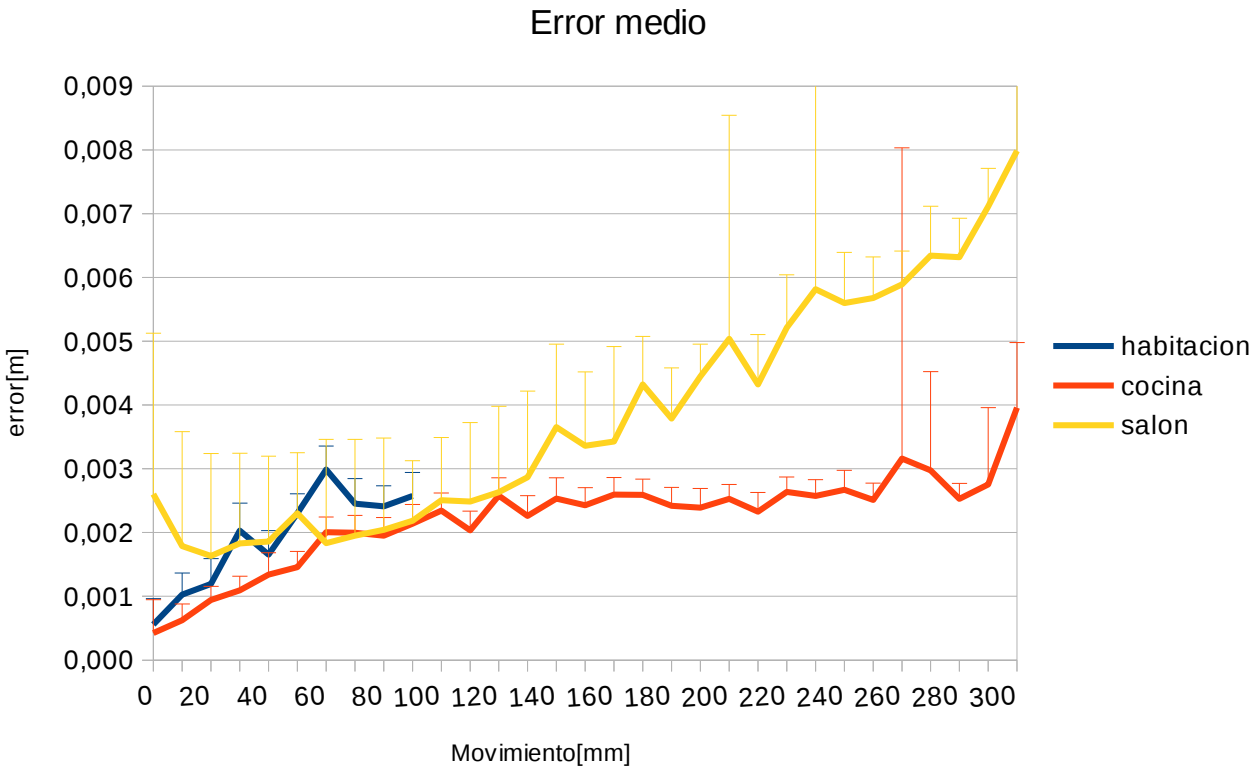
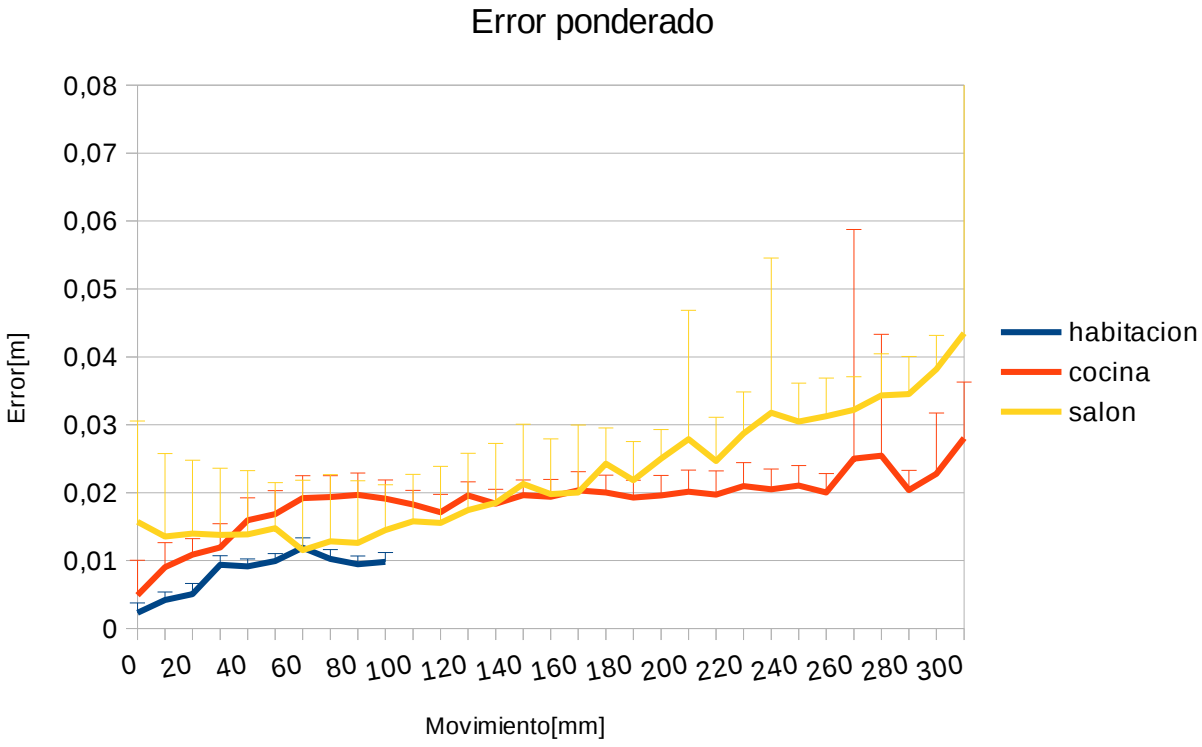
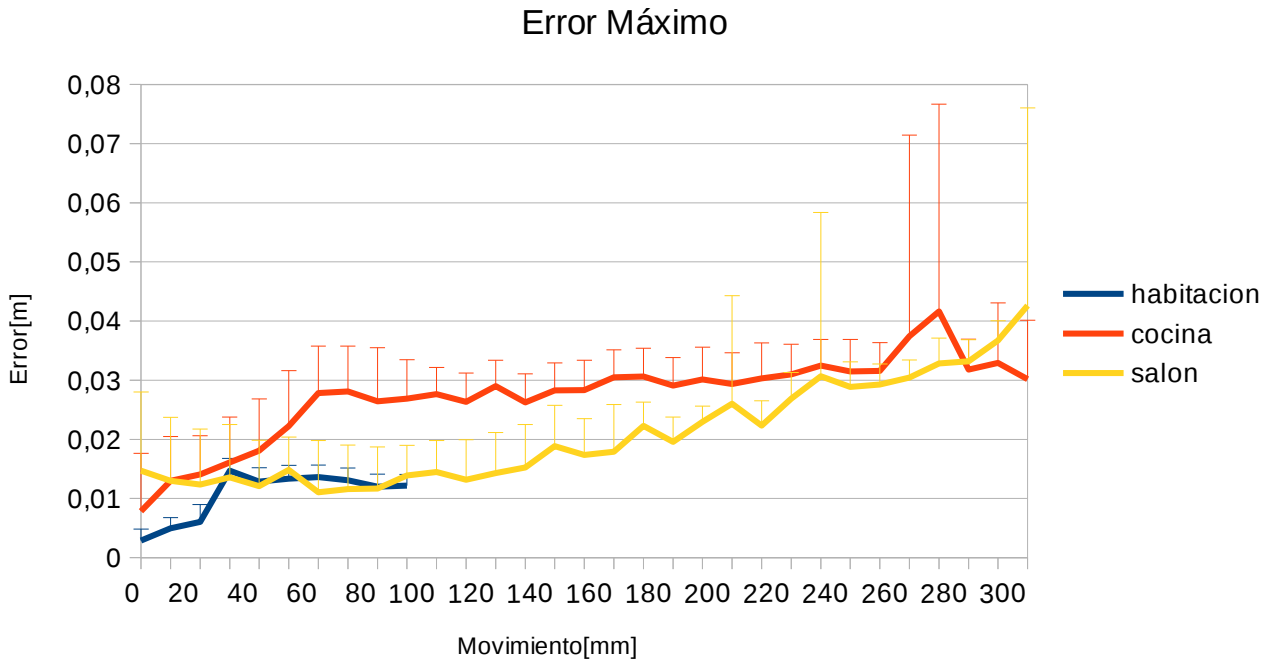


Figura 6.12: esquema de registro de nubes con procesos paralelos

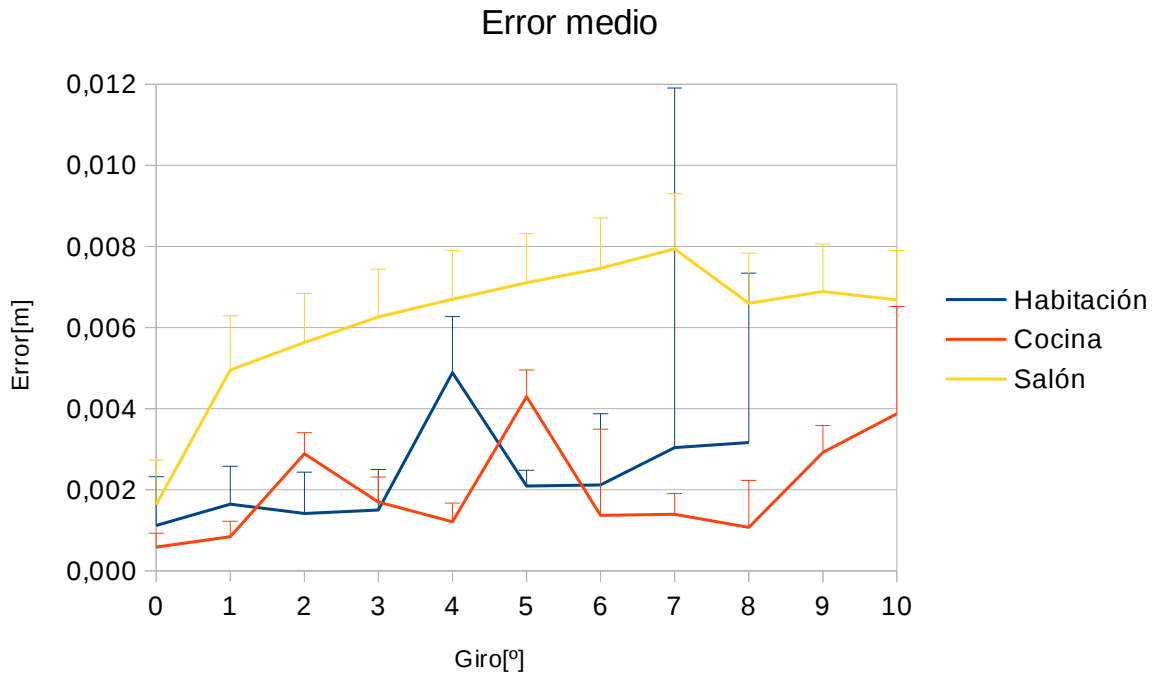
Gráficas de movimiento rectilíneo:

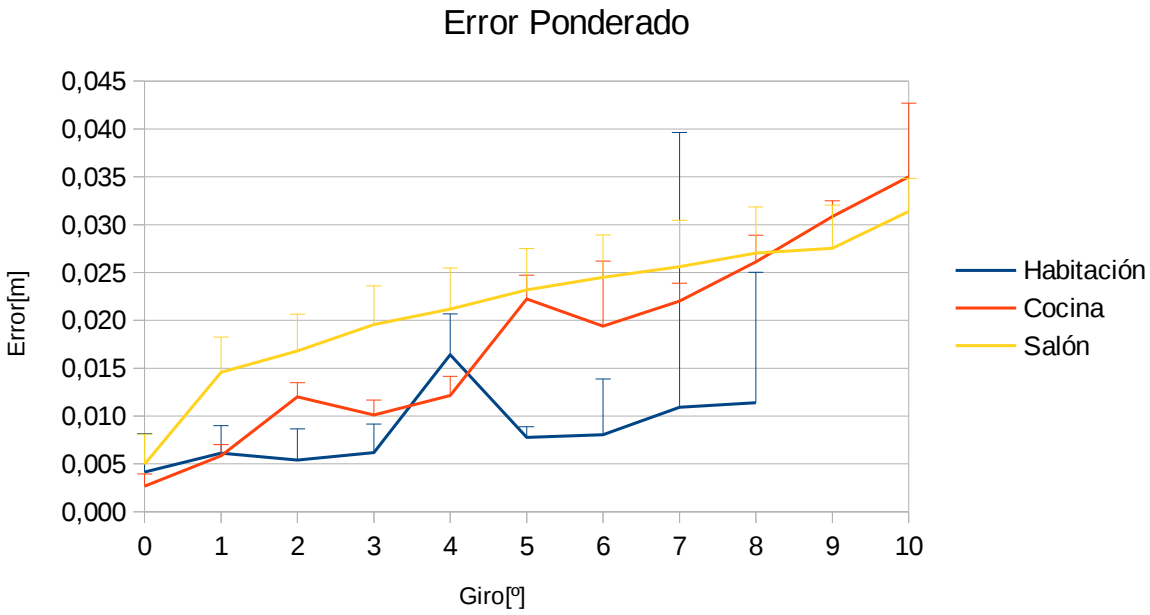

Error medio
0,009
0,008
0,007
0,006
0,005
0,004
0,003
0,002
0,001
0,000
error[m]
0 20 40 60 80 100 120 140 160 180 200 220 240 260 280 300
Movimiento[mm]
habitacion
cocina
salon

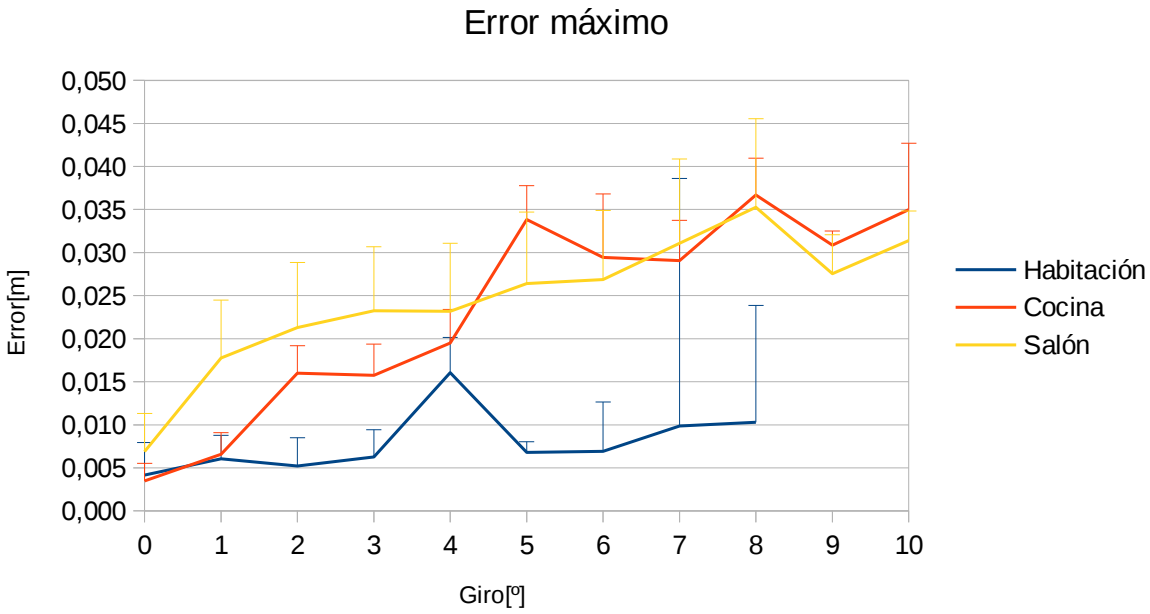

Error ponderado
0,08
0,07
0,06
0,05
0,04
0,03
0,02
0,01
0
Error[m]
0 20 40 60 80 100 120 140 160 180 200 220 240 260 280 300
Movimiento[mm]
habitacion
cocina
salon



Gráficas de movimiento rotación.



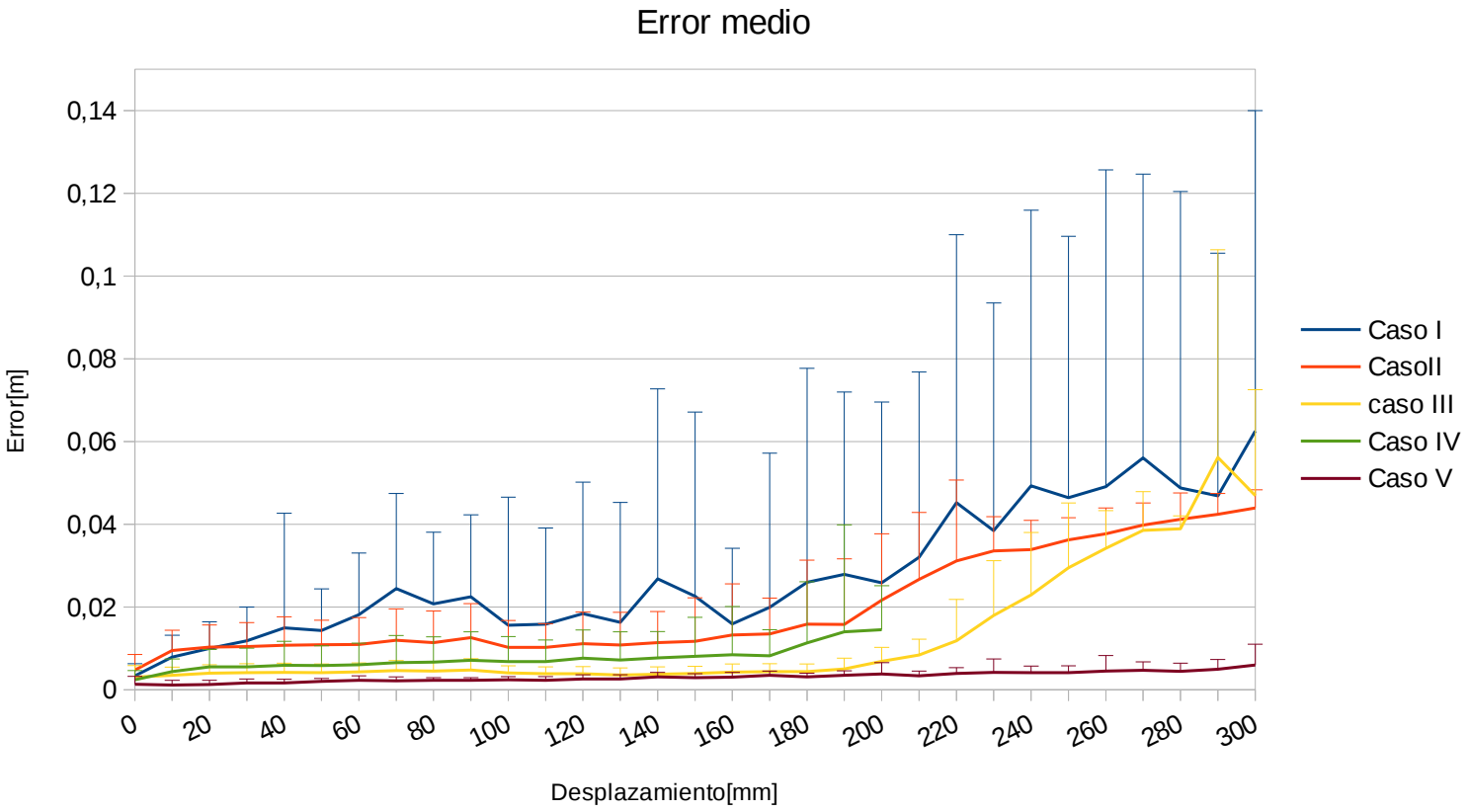


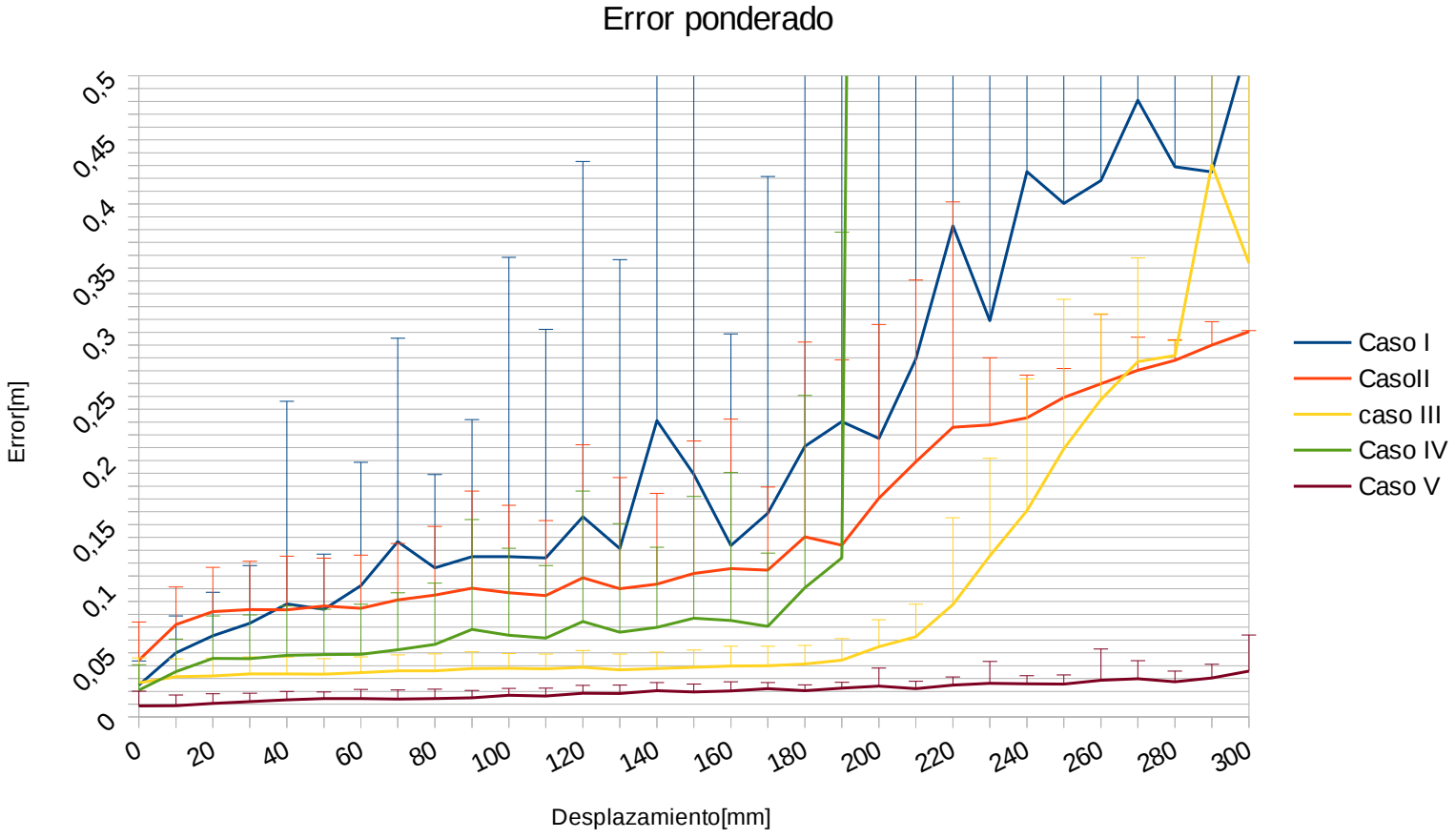


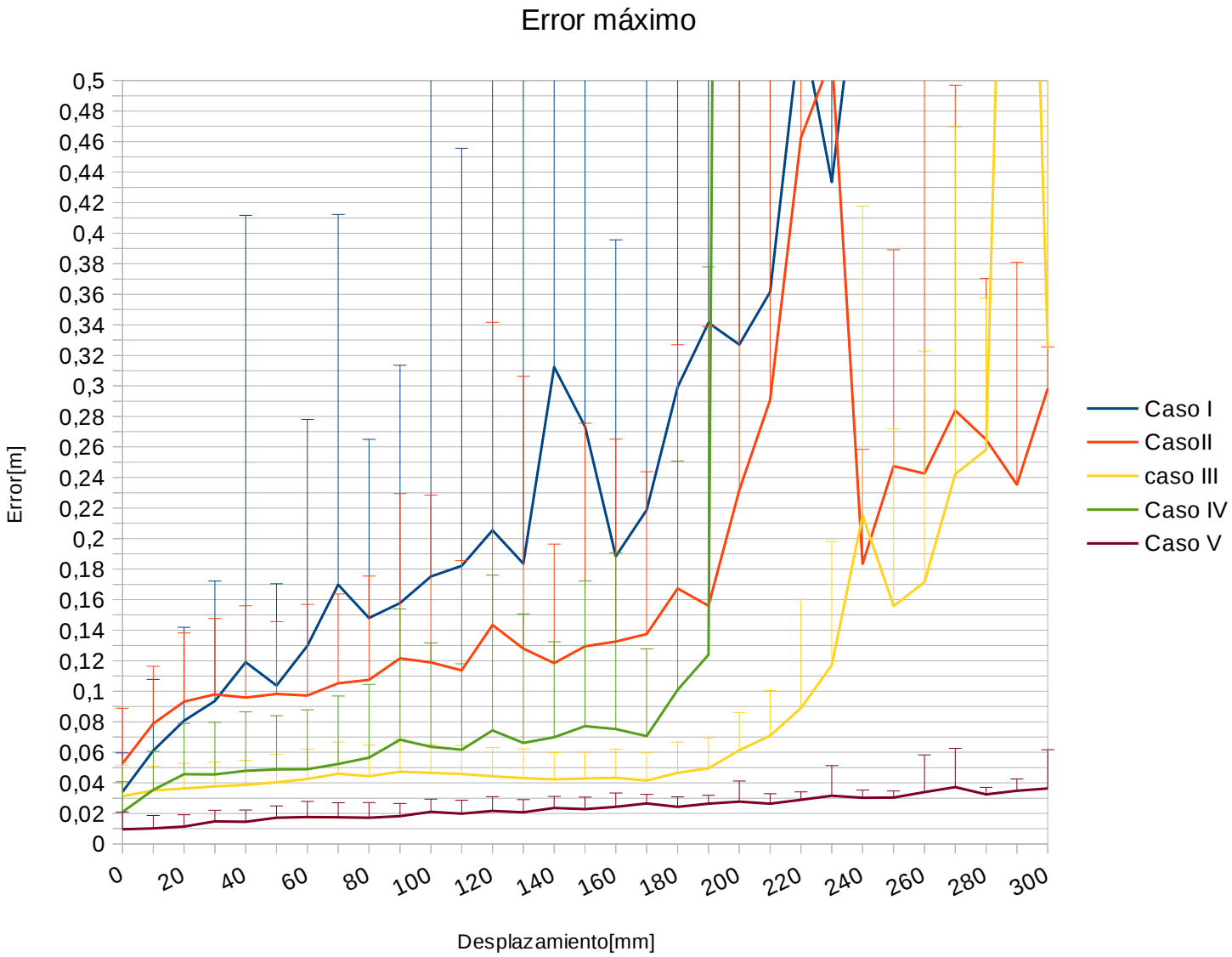
En las gráficas anteriores se puede demostrar cómo tanto en los movimientos lineales, como en el giro, el error si bien dobla o triplica el error inicial este es ínfimo en términos relativos. Por otra parte, cabe destacar como el error se mantiene acotado de forma estable dentro de unos valores, así como la desviación del error es pequeña con valores similares al de los valores iniciales, lo que resulta eficaz para ser utilizado en casos prácticos.

En los siguientes gráficos se realizarán una comparación de los diferentes métodos estudiados hasta este punto.

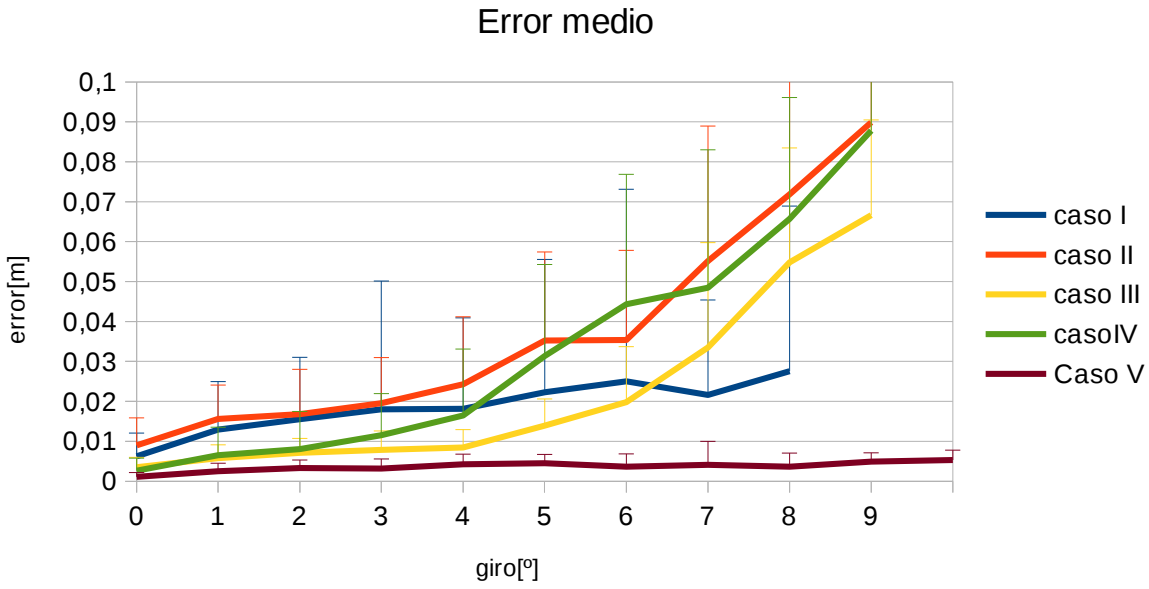
Gráficas de movimiento rectilíneo:

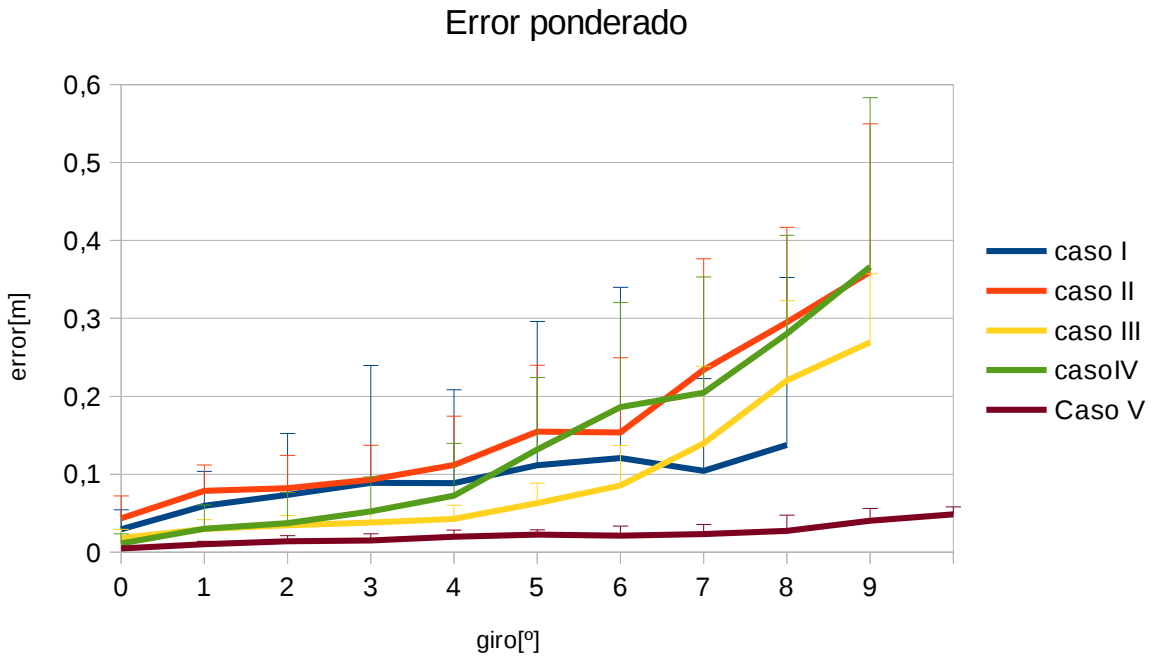


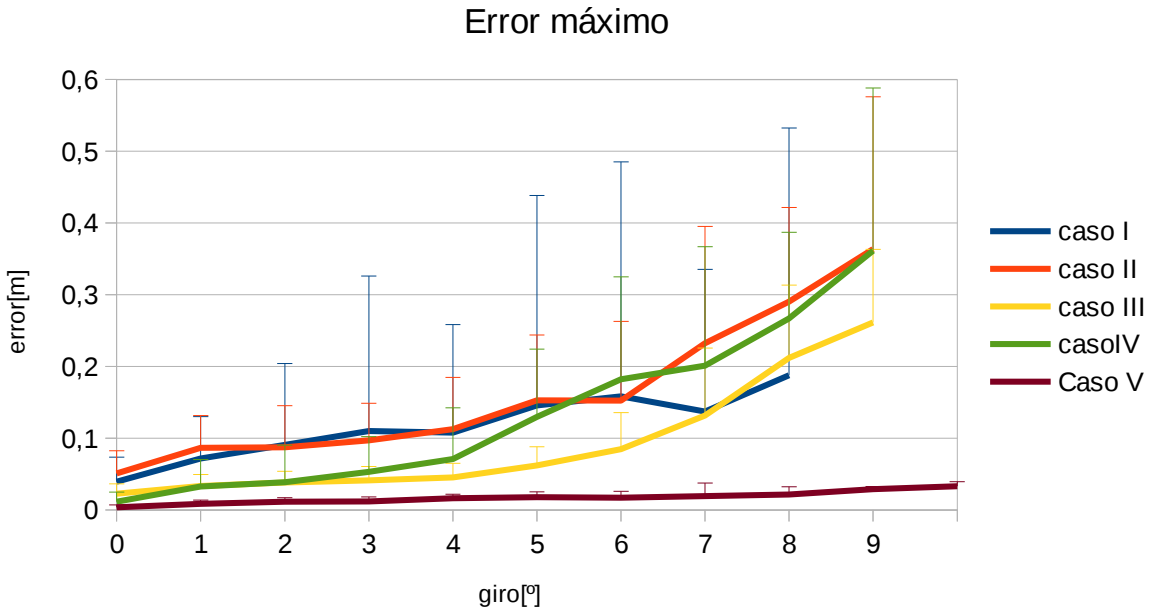




Gráficas de movimiento rotación.







De las diferentes gráficas, tanto en los desplazamientos lineales y los giros, es 'caso V' mantiene una mayor estabilidad en el error, tanto por mantener un error relativamente bajo con respectos a los demás casos, como por mantener acotada la desviación del error.

Para demostrar la calidad del método multietapa representado en el 'caso V', mostramos la siguientes figuras donde se muestra el registro final de un muestra de varias nubes de puntos.

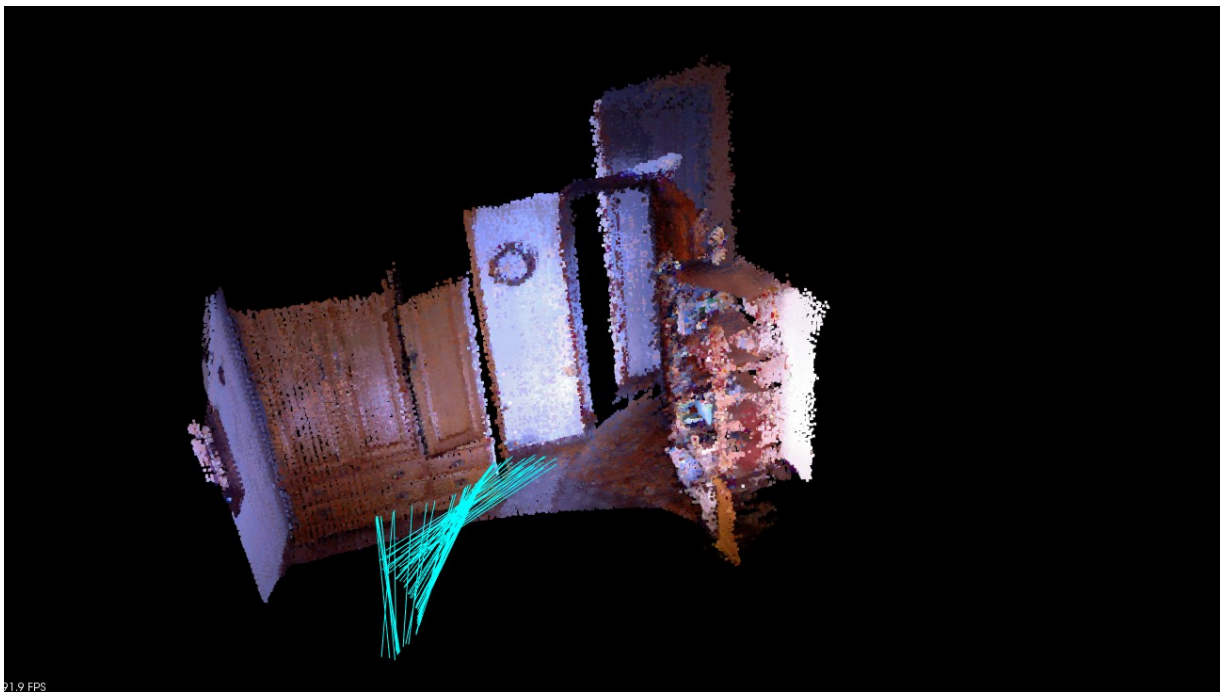


Figura 6.13: registro de más de 50 nubes de puntos del interior de una habitación con la posición y orientación mediante líneas de cada nube de puntos.

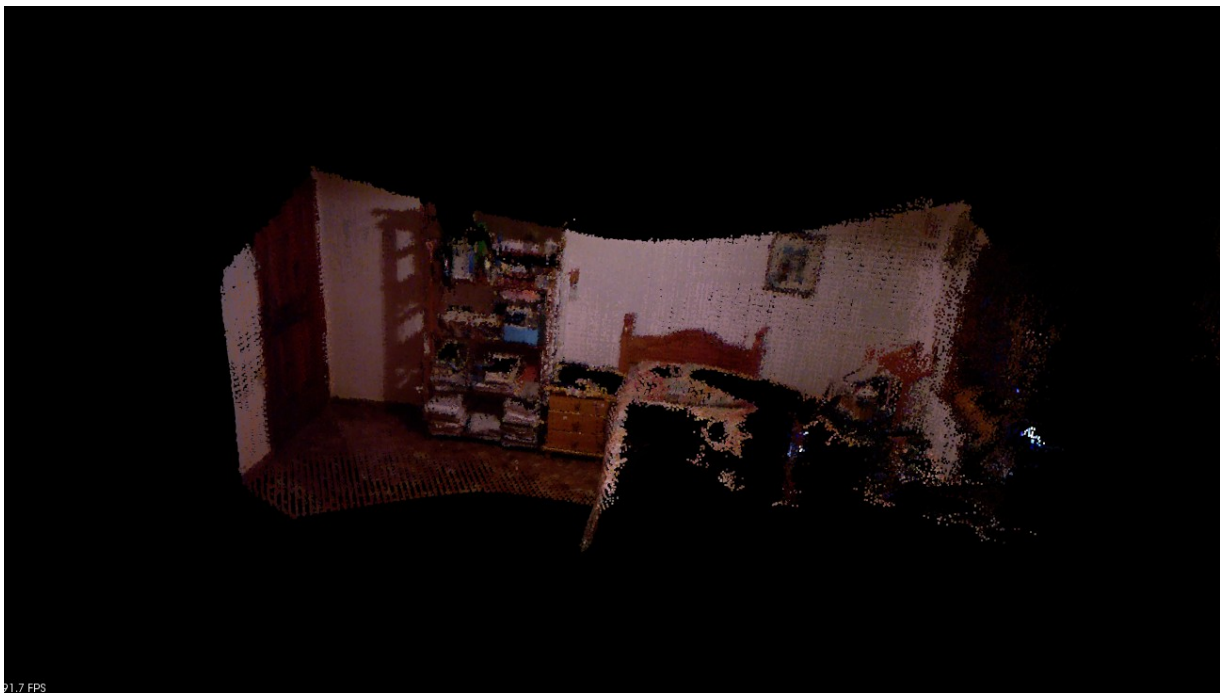


Figura 6.14: registro de más de 50 nubes de puntos del interior de una habitación.

7 Conclusiones

Los resultados de los diferentes estudios muestran que no es posible realizar un registro eficaz mediante una etapa. Para que tenga robustez es necesario contar con varios métodos, que nos permitan un acercamiento inicial de forma rápida. También es necesario tener una etapa de refinado, donde en nuestro estudio hemos aplicado el algoritmo de ICP por aplicar un error máximo de correspondencias que se considera más eficaz.

Sin embargo, no hemos conseguido que el algoritmo sea lo suficientemente rápido como para funcionar en tiempo real para que pueda un sistema moverse lo suficientemente eficaz como para desarrollar tareas. Se han implementado los algoritmos de forma paralela en la medida que la secuencia de tareas a realizar lo permite, mediante la clase multihilo de Boost.

La ineficiencia es debida a los elevados tiempo de cómputo que suponen los procesos de estudios de puntos, tanto obtención de *keypoint*, *feature* o RANSAC. En el anexo III se desarrolla como se comportan los tiempos de cómputo de varias clases de la librería y cómo afecta el volumen de datos. Si bien podría atenuarse con la aplicación de filtros de reducción del tamaño de la nube, pero esto haría que mucha información dejará de estar disponible. Por tanto, no sería suficiente para conseguir el objetivo.

Por lo tanto, contamos con algoritmos 3D que están en seria desventaja con los sistemas tradicionales de percepción 2D, ya que estos tienen algoritmos más sencillos y rápidos de ejecutar.

En este proyecto hemos tenido algunas dificultades para poder extraer la información de las nubes de puntos, pues existen muchos errores de medida en las capturas de Kinect. Una línea a seguir para solucionar este problema sería tener en cuenta los algoritmos que se utilizan en odometría 2D, mediante los cuales se verificaría que los puntos son consistentes y se recupera parte de lo que no tienen la información de profundidad.

Algunas cámaras, como la kinect, disponen de otros sensores que pueden medir el movimientos y los giros que en esta se producen, si estos datos fueran accesibles no serían necesarios algunos de los pasos que más tiempo de cómputo requieren, como la búsqueda de keypoint. Aunque en nuestro proyecto solo queríamos trabajar con la imagen, hay que reconocer que los sensores que se requieren para obtener esta información son muy baratos, pero se debe tener en cuenta que los muestreos coincidan con la toma de la imagen correspondiente.

El enfoque de registrar líneas y planos para realizar un primer acercamiento, podría servir de línea de estudio para el registro de nubes mediante este método de forma definitiva. También es posible hacerlo mediante correspondencia de formas geométricas como cubos o esferas (aunque puede suceder que sean escasos).

No siempre es posible disponer un espacio donde los elementos que lo conforman están en

reposo, o es necesario interactuar mientras se reconoce el medio y sus cambios. Por esto, la siguiente línea de investigación a seguir es la de realizar mapeos, o reconocer el medio en ambientes con sólidos que se mueven como serían personas, animales, máquinas, árboles y objetos con movimientos restringidos, con el fin de reconocerlos y darles un tratamiento diferente al resto de la escena.

En cuanto a la librería PCL es necesario disponer de algoritmos que no requieran tanto tiempo de cómputo para filtrar una nube de puntos y obtener los keypoint. Si bien la repetibilidad de los keypoints es aceptable, se debería de investigar mejoras que sin tener que realizar tratamientos previos que en muchos casos requieren más tiempos y no supone una mejora importante.

8 Presupuesto

En este capítulo se detallará los gastos realizados para la elaboración de este proyecto, teniendo en cuenta la legislación vigente.

8.1 Hardware

Los costes de las herramientas necesarias para realizar este proyectos han consistido:

Elemento	Descripción	Precio
Equipo	Se compone de un procesador Intel® Core™ i7-4770 CPU @ 3.40GHz × 8 y elementos necesarios para su funcionamiento	750€
Pantalla	HP Compaq LA1951q	200€
Periféricos	Teclado, raton, sonido.	40€
Sistema de percepción	Kinect® para Xbox 360 con fuente de alimentación compatible con PC	150€

Como todas estas herramientas de Hardware solo han sido utilizadas durante un período de amortización, en la Ley 27/2014, de 27 de noviembre, del Impuesto sobre Sociedades, se establece el cálculo, los períodos y tipos de amortización según las características del bien amortizado, en la siguiente tabla se calculará la amortización total de los equipos en el proyecto:

Elemento	Precio	Amortización	Total amortizado
Equipo	750€	25%	187.5€
Pantalla	200€	25%	50€
Periféricos	40€	25%	10€
Sistema de percepción	150€	20%	30€
			227,5€

8.2 Documentación

Para poder desarrollar el proyecto así como comprender algunos de los algoritmos desarrollados en PCL, ha sido necesario realizar la adquisición de documentos .

Adquisición de 15 documentos científicos, con coste de 35€ total ….....525€

8.3 Personal

Para el cálculo de los costes de personal se tiene en cuenta la legislación española, para establecer los salarios se tiene en cuenta el *"XVI Convenio colectivo estatal de empresas de consultoría y estudios de mercado y de la opinión pública"*, aprobado en Boletín Oficial Del Estado, donde se incluye las actividades desarrollo de hardware y software. En el convenio se divide los salarios mínimos por categorías profesionales, según la características del puesto y la titulación requerida. Se desglosa en la siguiente tabla las tareas y horas de trabajo.

Puesto	Tarea	Grupo
Titulado de Grado Superior	Responsable del proyecto	Grupo I
Analista-programador	Programación en lenguaje C++	Grupo II

Es necesario tener en cuenta los gastos a realizar según los artículos *19 y 148 de Real Decreto Legislativo 8/2015, de 30 de octubre, por el que se aprueba el texto refundido de la Ley General de la Seguridad Social*, se establece que las bases de cotización. Estas son aprobadas en el a*rtículo 115* de la *Ley 48/2015, de 29 de octubre, de Presupuestos Generales del Estado para el año 2016.* El grupo de cotización se establecerá según el convenio.

En la siguiente tabla se calcula el gasto en salarios por una empresa, teniendo en cuenta la cotización por contingencias comunes calculado por las tablas de la seguridad social es del 30.5%:

Puesto	Salario Base mensual(x14)	Salario base+Cotización(x12)
Titulado de Grado Superior	1496,74€	2278,78€
Analista-programador	1468,54	2235,82€

La duración del proyecto ha sido de seis meses, en la siguiente tabla se aplican las horas de trabajo así como el coste total de cada trabajador:

Puesto	Salario base+Cotización(x12)	Meses de trabajo	Coste total
Titulado de Grado Superior	2278,78€	1	2278,78€
Analista-programador	2235,82€	5	11179,10€
			13457,88€

8.4 Coste totales

Tipo coste	Coste total
Hardware	227,5€
Documentación	525€
Personal	13457,88€
	14,210.38 €

9 Referencias

[1] K. S. Arun, T. S. Huang, and S. D. Blostein, “Least squares fitting of two 3-D point sets,” IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), vol. 9, no. 5, pp. 698–700, 1987.

[2] D. W. Eggert, A. Lorusso, and R. B. Fisher, “Estimating 3-D rigid body transformations: a comparison of four major algorithms,” Mach. Vision Appl., vol. 9, no. 5-6, pp. 272–290, 1997.

[3] Y. Chen and G. Medioni, “Object modelling by registration of multiple range images,” Image Vision Comput., vol. 10, no. 3, pp. 145–155, 1992.

[4] A. W. Fitzgibbon, “Robust registration of 2D and 3D point sets,” in British Machine Vision Conference (BMVC), pp. 411–420, 2001.

[5] K. L. Low, “Linear least-squares optimization for point- to-plane ICP surface registration,” in Technical Report TR04-004, Department of Computer Science, University of North Carolina at Chapel Hill, 2004.

[6] S. Rusinkiewicz and M. Levoy, “Efficient variants of the ICP algorithm,” in Proc. Int’l Conf. 3D Digital Imaging and Modeling, pp. 145-152, 2001.

[7] J. Diebel, K. Reutersward, S. Thrun, J. Davis, and R. Gupta, “Simultaneous localization and mapping with active stereo vision,” in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3436–3443, 2004.

[8] C. V. Nguyen, S. Izadi, and D. Lovell, “Modeling Kinect sensor noise for improved 3D reconstruction and tracking,” in 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), pp. 524 –530, 2012.

[9] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, “RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments,” International Journal of Robotics Research (IJRR), vol. 31, no. 5, pp. 647–663, 2012.

[10] A. Segal, D. Haehnel, and S. Thrun. “Generalized-icp”. In Proceedings of Robotics: Science and Systems, Seattle, USA, June 2009.

[11] M. Potmesil. “Generating Models for Solid Objects by Matching 3D surface Segment,” In Proceeding of the international Joint Conference on Artificial Intelligence, pp. 1089-1093, 1983.

[12] R. B. Rusu and S. Cousins “3D is here: Point Cloud Library (PCL)”,IEEE International Conference on Robotics and Automation (ICRA), pp. 1-4,2011.

[13] C. Dal Mutto, P.Zanuttigh, G. Cortelazzo, “Time-of-Flight Cameras and Microsoft Kinect”.Springer-Verlag New York, USA, 2013.

[14] K.L. Low, “Linear least-squares optimization for point- to-plane ICP surface registration,” , Reporte técnico de la Universidad de Carolina del norte en Chapel Hill, 2004.

[15] J. Wang, R. Lindenbergh, Y. Shen and M. Menenti, “Coarse Point Cloud Registration by EGI Matching of Voxel Clusters” In ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 97-103, Volume III-5, 2016.

[16]B. Steder, R. B. Rusu, K. Konolige and W. Burgard,”Point feature extraction on 3D range scans taking into account object boundaries.” IEEE International Conference on Robotics and Automation ICRA,pp.2601-2608, 2011.

[17]S. M. Smith and J. M. Brady, “SUSAN – A new approach to low level image processing,” International Journal of Computer Vision , vol. 23, no. 1, pp. 45–78, 1997.

[18]R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha and M. Beetz. “Towards 3D Point Cloud Based Object Maps for Household Environments” Robotics and Autonomous Systems Journal,pp. 927-941, vol 56, 2008

[19] R. B. Rusu “Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments” Thesis at Computer Science department, Technische Universitaet Muenchen, 2009.

[20] E. Mair, G. D. Hager, D. Burschka, M. Suppa and G. Hirzinger “Adaptive and Generic Corner Detection Based on the Accelerated Segment Test"In Proceedings of the 11th European Conference on Computer Vision (ECCV),pp. 183-196, Part II ,2010.

[21]http://mathworld.wolfram.com/EulerAngles.html fecha : septiembre de 2016

Anexo I

En este anexo se explicarán las diferentes tecnologías que existen para la obtención de capturas de imágenes 3D o nube de puntos. También veremos los principios físicos que se han desarrollado según Dal Mutto[14] y se mostrarán algunos de los productos que recomienda la librería PCL para la adquisición de datos en su formato.

1 Time-of-flight

Un sensor de Time-of-Flight(ToF) calcula su distancia radial desde un punto de la escena por el principio de tiempo de vuelo o RADAR(Radio Detection And Ranging). Como la radiación electromagnética viaja en el aire a la velocidad de la luz $c \approx 3*10^8$[m/s], la distancia ρ recorrida en un tiempo τ por una radiación óptica es $\rho = c\tau$, la figura AI.1 muestra el esquema de medición típica de ToF, la radiación emitida en el tiempo 0 por el transmisor del sensor ToF de la izquierda se desplaza en línea recta hacia el escenario para una ρ distancia, se refleja en la superficie y vuelve desplazándose de nuevo una distancia ρ y en el tiempo τ alcanza el receptor del sensor ToF, idealmente tendría la misma posición que el transistor, dado que en el momento τ la longitud del camino cubierto por la radiación es 2ρ, es la base de las mediciones en cámaras de ToF.

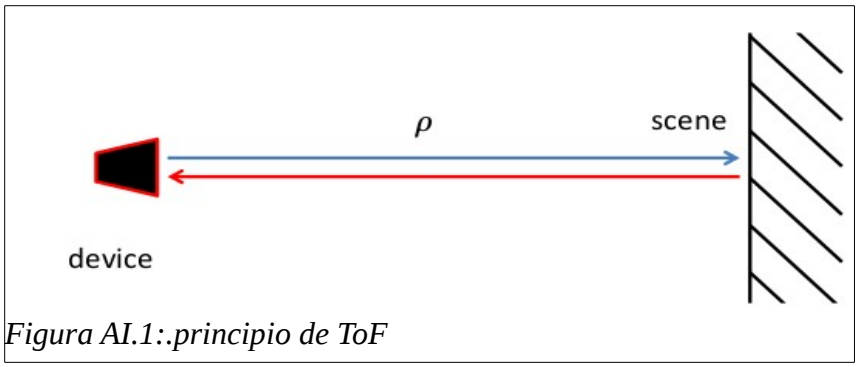


Figura AI.1:.principio de ToF

A pesar de la simplicidad conceptual, su aplicación esconde enormes desafíos tecnológicos, ya que hay que tener en cuenta la velocidad de la luz. Por ejemplo, para cubrir una distancia de 1mm tardaría 5ps la luz en recorrer dicha distancia, es necesario sensores que sean capaces de registrar pasos de tiempo de al menos 5ps. La implementación de este tipo de dispositivos y su integración en configuraciones matriciales son cuestiones fundamentales del desarrollo de sistemas ToF actual. Diferentes opciones tecnológicas de reloj conducen a diferentes tipos de cámaras ToF.

Swiss Ranger 4000

La SR4000 es la 4ª generación de cámaras de Tiempo de Vuelo (TOF-Time Of Flight) de MESA. Proporcionan una información de distancia estable en un formato robusto y reducido de 65x65x68mm.

El software de visión HALCON de MVTEC proporciona un nuevo driver que permite la captura de forma directa con estas cámaras. Esta nueva interfaz permite una fácil adquisición de datos 3D, tales como distancia, amplitud e imágenes en los planos cartesianos X/Y/Z, basada en la tecnología Time-of-flight. Dicho interfaz permite acceder a los parámetros de control de cámara tales como (shutter, gain, threshold…) generando un mapa de profundidad de 16 bits listo para su procesamiento con algoritmos de visión integrados en HALCON. Mediante este nuevo interfaz disponemos de forma directa de mapas de profundidad en tiempo real totalmente calibrados y listos para su posterior procesado sin necesidad de complejas manipulaciones de las nubes de puntos para la extracción de la información deseada.

Como características más destacables cabe citar:

• Cámara de grado industrial, proporciona medidas de alta calidad en entornos donde se requiere alto rendimiento. Esto se consigue mediante el diseño esmerado de los componentes de la cámara así como a los procesos de calibración realizados en fabrica. IP40. Certificada CE: iluminación clase 1, EMC clase A.

• Medidas estables y continuas garantizadas mediante el diseño óptico de autocalibración.

• Medidas independientes del color y reflectividad del objeto. Se consiguen medidas estables en repetibilidad y precisión incluso en objetos de distintos colores y reflectividad dentro de la misma imagen.

• Se dispone de cámaras de dos rangos de medidas de 5m y 10m permitiendo más flexibilidad y precisión en la medida si se conoce la distancia a la que se medirán los objetos.

• Posibilidad de hacer medidas con multicámaras mediante la utilización de múltiples frecuencias de iluminación (hasta 3 cámaras)

• Supresión de luz de fondo en píxel permite medidas de mayor precisión incluso en condiciones de luz de fondo desfavorable.

• Modo de adquisición seleccionable facilita las medidas en modo continuo o mediante trigger. El modo trigger es muy útil para sincronizar la cámara con otros dispositivos.

Figura AI.2: sensor MESA SR4000 cortesía de hptg.com

2 Modelo Pin-Hole

Considerando un sistema de referencia 3D (con los ejes x,y,z), llamado sistema de coordenadas de la cámara, y un plano paralelo al plano (x,y) intercepta el eje z una coordenada negativa f, llamada sensor o imagen plana S como se muestra en la Figura AI.3. La orientación de los ejes sigue la convención de la regla de la mano derecha. Considere un sistema de referencia 2D asociado al sensor, al que nos referiremos como S-2D, orientado como se muestra en la figura AI.3. La intersección c del eje z con el plano del sensor tiene coordenadas p = [u = cx , v = cy]^T, el conjunto de puntos p del sensor, llamados píxeles, de coordenadas p=[u,v]^T obtenidos desde la intersección de los rayos que conectan el centro de proyección O con todos los puntos P de la escena 3D con coordenadas P=[x,y,z]^T es la huella de la escena en el sensor S. La relación entre P y p, es llamada proyección central o en perspectiva, como se muestra fácilmente por similitud de triángulos, donde la distancia |f| entre el plano del sensor y el centro de proyección O es llamado distancia focal.

$$u - c_x = f\frac{x}{z}$$
$$u - c_y = f\frac{y}{z} \qquad \text{(AI.1)}$$

La proyección en perspectiva es una buena descripción, de la relación geométrica entre las coordenadas de los puntos de la escena y los de la imagen obtenidos por el dispositivo de proyección pin-hole, con el centro de proyección O. Este sistema permite que un solo rayo de luz traspase el agujero del sistema pin-hole en O. Por una serie de razones, en los sistemas de imagen es más práctico utilizar la óptica. Es decir, conjunto de lentes adecuados en lugar del sistemas pin-hole. Muy notablemente el modelo ideal de una óptica, llamado modelo de lente fina, mantiene la relación entre las coordenadas de P y p si el centro óptico es en O y el eje óptico de la lente. La línea de intersección ortogonal de la lente en su punto nodal, es ortogonal al sensor. Si una lente delgada sustituye a un sistema pin-hole en la figura AI.3.c, el eje óptico coincide con el eje z del sistema de coordenadas de la cámara.

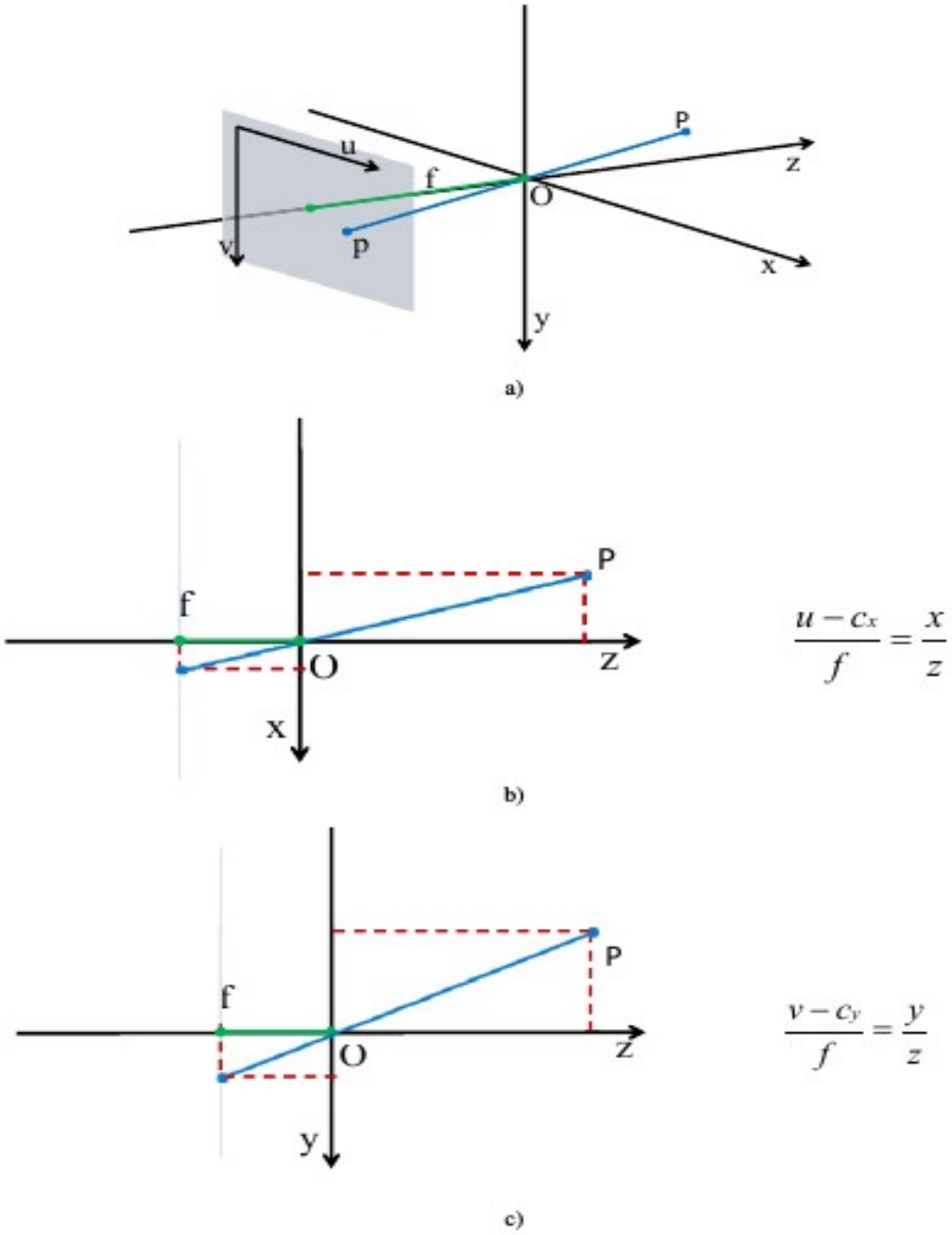


Figura AI.3: esquema del modelo Pin-Hole

Parámetros intrínsecos y extrínsecos de la cámara

La geometría proyectiva asociada a cada punto p en 2D con coordenadas cartesianas $p=[u,v]^T$ de un plano en representación 3D, llamada coordenadas homogéneas $\tilde{p}=[hu,hv,h]^T$, donde la h es una constante real. El uso de h = 1 es bastante común y $[u, v, 1]^T$ es a menudo llamado el vector extendido de p.

Las coordenadas de $p=[u, v]^T$ pueden ser obtenidas mediante $\tilde{p}=[hu,hv,h]^T$ dividiéndolos por la tercera coordenada h. El vector $\tilde{p}$ puede ser interpretado como el rayo que conecta el punto 3D con el centro de proyección O y el punto de proyección del sensor p.

De manera similar cada punto P con coordenadas cartesianas $P=[x,y,z]^T$ puede ser representada en coordenadas homogéneas por un vector 4D $\tilde{p}=[hu,hv,hz,h]^T$ donde h es cualquier constante real. El vector $[x,y,z,1]^T$ es llamado el vector extendido de P. Las coordenadas de $P=[x,y,z]^T$ puede ser obtenido desde $\tilde{p}=[hu,hv,hz,h]^T$ dividiéndolos por su cuarta coordenada h. Las coordenadas homogéneas representadas en p acepta para nuevas versiones con relaciones no lineales, como el modelo pin-hole, una conveniente forma matricial, como :

$$z\begin{bmatrix}u\\v\\1\end{bmatrix}=\begin{bmatrix}f & 0 & c_x\\0 & f & c_y\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}x\\y\\z\end{bmatrix} \qquad \text{(AI.2)}$$

El lado izquierdo de la expresión AI.2 representa las coordenadas homogéneas de p en 2D, pero el lado derecho de AI.2 representa las coordenadas cartesianas de P en 3D. Para representar las coordenadas homogéneas de se agrega una columna de 0 a la derecha de la matriz.

Los dispositivos de sensores digitales son típicamente matrices planas de células con sensores rectangulares alojados en el sistema de conversión fotoeléctrica, de tecnología CMOS o CCD, en el caso de las cámaras de vídeo, foto. Habitualmente se modelan como una rejilla rectangular ΛS con tamaño de paso rectangular y vertical k_u y k_v, respectivamente como se muestra en la figura AI.4.

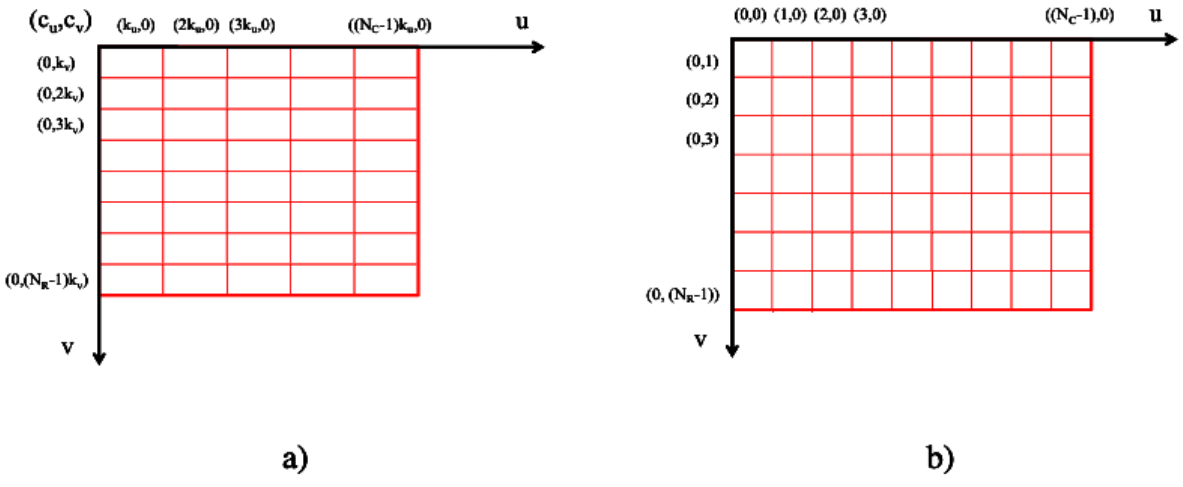


Figura AI.4: ejemplo de matriz plana de un sensor CCD o CMOS

Teniendo en cuenta el tamaño finito del sensor, con solo una ventana rectangular ΛS compuestas por NC columnas y NR filas es de interés para la composición de la imagen. Con el fin de hacer frente al enrejado con origen (0,0) y coordenadas de píxel unitarias $u_S \in [0, ..., N\,C - 1]$ y $v_S \in [0, ..., NR - 1]$, tanto en la dirección u y v, la relación de coordenadas homogéneas matricial es reemplazado por:

$$z\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K\begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{AI.3}$$

donde K es la matriz de parámetros intrínsecos definidos como

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{AI.4}$$

con $f_x = fk_u$ el eje x longitud focal de la óptica, $f_y = fk_v$ el eje y longitud focal de la óptica,cx y cy las coordenadas (u,v) de la intersección del eje óptico con el plano del sensor. Todas estas cantidades están expresadas en [píxel], es decir, f está en [mm], k_u y k_v se supone que son [píxel] / [mm].

En muchas situaciones prácticas, no es conveniente representar los puntos 3D de las escena con respecto al sistema de coordenadas de la cámara, sino con respecto a un sistema de referencia de fácil acceso y diferente, llamado convencionalmente sistemas de coordenadas del mundo. En el que un punto es denotado como P_w que tiene las coordenadas $P_w = [x_w, y_w, z_w]^T$. La relación entre la representación de un punto de la escena con respecto al sistema de coordenadas de la cámara, que se denota como P, y su representación con respecto al sistema de coordenadas del mundo denotado P_w es

$$P = R\,P_w + t \tag{AI.5}$$

donde R es la matriz de rotación y t es el vector de traslación.

Al representar P_w en lado derecho de las coordenadas homogéneas $\widetilde{P_w} = \left[hx_w, hy_w, hz_w, h\right]^T$ y eligiendo h=1, la relación puede ser escrita como:

$$P = \begin{bmatrix} R & t \end{bmatrix}\widetilde{P_w} \tag{AI.6}$$

En este caso la relación entre la representación del punto de la escena en coordenadas homogéneas con respecto al sistema de coordenadas del mundo y su píxel correspondiente en coordenadas homogéneas también, la ecuación AI.3 se convierte en:

$$z\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = KP = K\begin{bmatrix} R & t \end{bmatrix}\widetilde{P_w} = M\widetilde{P_w} = M\begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \qquad \text{(AI.7)}$$

donde la matriz 3 x 4

$$M = K\begin{bmatrix} R & t \end{bmatrix} \qquad \text{(AI.7)}$$

es llamada matriz de proyección. La matriz M depende de la matriz de parámetros intrínsecos K y de las matrices de parámetros extrínsecos R y t del sistema de la imagen.

Como consecuencia de distorsiones y aberraciones de la óptica reales, las coordenadas $\hat{p} = (\hat{u}, \hat{v})$ del píxel en realidad asociada a la escena del punto P con coordenadas P = [x, y, z]^T en el sistema de coordenadas de la cámara no satisfacen la relación de la ecuación AI.3. Las coordenadas correctas del píxel (u,v) de la ecuación AI.3 puede ser obtenida desde las coordenadas distorsionadas realmente medidas $(\hat{u}, \hat{v})$, por el sistema de formación de imágenes por inversión de modelos de distorsión adecuados, es decir, $p_T = \Psi^{-1}(\hat{p}_t)$ donde Ψ denota la transformación de distorsión.

El modelo de antidistorción(z), también llamado modelo Heikkila, se ha hecho popular, ya que es adecuado para las distorsiones de la mayoría de los sistemas de imagen y el cálculo de sus parámetros :

$$\begin{bmatrix} u \\ v \end{bmatrix} = \Psi^{-1}(\widehat{p_T}) = \begin{bmatrix} \hat{u}\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) + 2 d_1 \hat{v} + d_2\left(r^2 + 2\hat{u}^2\right) \\ \hat{v}\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) + d_1\left(r^2 + 2\hat{v}^2\right) + 2 d_2 \hat{u} \end{bmatrix} \qquad \text{(AI.8)}$$

donde $r = \sqrt{(\hat{u} - c_x)^2 + (\hat{v} - c_y)^2}$, los parámetros k_i con i=1,2,3 son constante que representa la distorsión radial, y d_i con i=1,2 para la distorsión tangencial. También hay otros modelos de distorsión más complejos.

Parámetros de distorsión:

$$d = \begin{bmatrix} k_1, k_2, k_3, d_1, d_2 \end{bmatrix} \qquad \text{(AI.8)}$$

son parámetros intrínsecos que deben ser considerados junto con [f,k_u,k_v,c_x,c_y]. La estimación de parámetros intrínsecos y extrínsecos del sistema de la imagen es llamado calibración geométrica.

3 Visión estereoscópica

Un sistema de visión estéreo está formado por dos cámaras estándar (normalmente idénticas) que enfocan parcialmente la misma escena. Estos sistemas siempre se pueden calibrar y rectificar. También es equivalente a un sistema de visión estéreo con cámaras idénticas estándares, con sensores de imágenes alineados y el eje óptico paralelo. La calibración de estos sistemas se realiza mediante la geometría equipolar.

Las dos cámaras del sistema de visión estéreo S tiene la cámara derecha L(cámara de referencia) y la cámara izquierda R (cámara objetivo). Estas cámaras tienen su propio sistema de coordenadas de la cámara 3D y sistema de referencia 2D. Para la cámara L tenemos sistema de coordenada (x_L,y_L,z_L), también llamado sistema de referencia L-3D, y un sistema de referencia 2D con coordenadas (u_L,v_L). La cámara R tiene sistema de coordenadas con coordenadas (x_R,y_R,z_R), también llamado sistema de referencia R-3D, y un sistema de referencia 2D con coordenadas (u_R,v_R). Convencionalmente se considera el sistema de referencia L-3D como sistema de referencia del sistema de visión estéreo, y para denotar este como sistema de referencia S-3D.

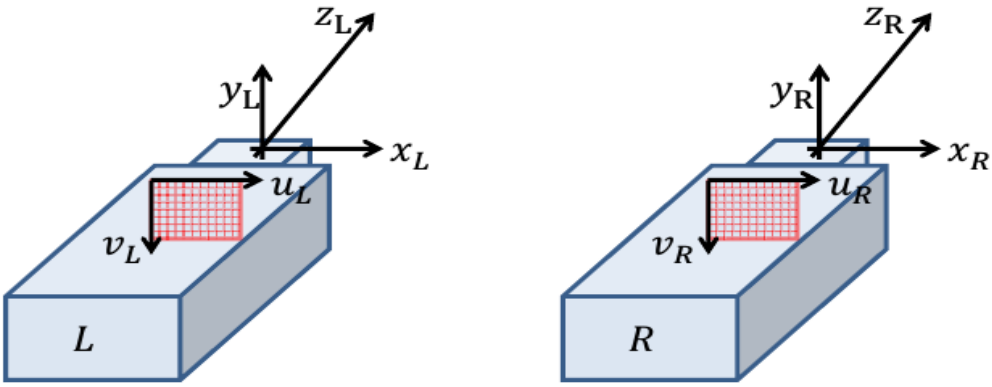


Figura AI.5: representación gráfica de sistema estereoscópica

Para el calibrado y rectificado de un sistema de visión estéreo, un punto P en 3D con coordenadas $P=[x,y,z]^T$ con respecto al sistema de referencia S-3D, es proyectado sobre el píxel p_L . P_R de la cámara L y R con coordenadas $p_L=[u_L,v_L]^T$ y $p_R=[\ u_R\text{-}d,v_R=v_L]^T$ respectivamente,como se muestra en la Figura AI.5. De las relaciones relativas al triángulo con vértices en la P_R, P y P_L se puede demostrar que en una configuración de rectificado, en puntos p_L y p_R tienen las mismas coordenadas verticales. La diferencia entre sus coordenadas horizontales $d = u_L - u_R$, llamada disparidad, es inversamente proporcional al valor de profundidad z de P, a través de la relación de triangulación conocida:

$$z=\frac{b|f|}{d} \qquad \text{(AI.9)}$$

Donde b es la línea de base, la distancia entre los orígenes de los sistemas de referencia L-3D y R-3D y | f | es la longitud focal de ambas cámaras. Los píxeles p_L y p_R se llaman conjugados. Desde las coordenadas 2D de p_L y la profundidad asociada z obtenida en la ecuación AI.9, las coordenadas x e y de los puntos 3D correspondientes, representado con respecto al sistema de coordenadas de la cámara, pueden ser computado por inversión de la ecuación de proyección relativa a la cámara L, de la siguiente forma:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = K_L^{-1} \begin{bmatrix} u_L \\ vL \\ 1 \end{bmatrix} z \qquad \text{(AI.10)}$$

Donde K_L^{-1} es la matriz inversa de los parámetros intrínsecos rectificados de la cámara L. Una vez una pareja de píxeles conjugados p_L y p_R, de la imagen estéreo, estén disponibles a partir de las coordenadas 3D del $P=[x,y,z]^T$ de los puntos de la escena, podemos computar mediante las dos relaciones anteriores, que normalmente es llamada triangulación o computación estereoscópica.

La disponibilidad de un par de píxeles conjugado es una parte difícil del procedimiento anterior. Porque un par de este tipo puede no existir debido a oclusiones, incluso si existe puede que no sea fácil encontrarla.

La detección de píxeles conjugados entre el par de imágenes estéreo, normalmente llamado problema de correspondencia, es uno de los principales desafíos de un algoritmo de visión estéreo. Los métodos propuestos para esta tarea se suelen dividir en los enfoques: locales y globales.

1. Métodos locales consideran solo las medidas de similitud cercanas entre la región circundante p_L y de forma similar regiones alrededor de todos los puntos conjugados candidatos p_R de la misma fila. El punto conjugado seleccionado es el que maximiza la medida de similitud, este es típicamente llamado estrategia Winner Takes All (WTA).

2. Los métodos globales no tiene en cuenta las parejas de puntos por sí mismo, sino que estima todos los valores de disparidad explorando esquema de de optimización global. Estos están basados en formulaciones bayesianas que recibien actualmente mucha atención. Estas técnicas generalmente modelan las escenas como un campo aleatorio de Markov(MKF), e incluyen dentro de un único marco de referencia, procedentes de las comparaciones locales entre las dos imágenes y las restricciones de suavidad de la profundidad de la escena. Los Algoritmos de Visión Global estéreo normalmente estiman, mediante minimización de una función de valores en la disparidad de la imagen con términos de datos, que representan los valores de encuentro local, similar a los algoritmos locales (por ejemplo, covarianza) y un término suavidad que define el nivel de de la disparidad de imagen que representa, por explícita o implícita discontinuidades.

Los algoritmos específicos pueden tener un impacto considerable en la solución del problema

de la correspondencia, la calidad final de la reconstrucción 3D estéreo depende inevitablemente de las características de la escena. Esto puede realizarse fácilmente teniendo en cuenta, el caso de una escena sin características geométricas o de color, como una pared recta de color uniforme. Las imágenes estéreo de una escena tales serán uniformes, puesto que no hay píxeles correspondientes que se pueden obtener a partir de ellas. Por lo que no hay información detallada acerca de la escena que puedan calcularse a través de la triangulación.

La triangulación activa utilizada en los llamados sistemas de codificación de luz introducidas, ofrece una forma eficaz de hacer frente a las cuestiones problemáticas correspondencia.

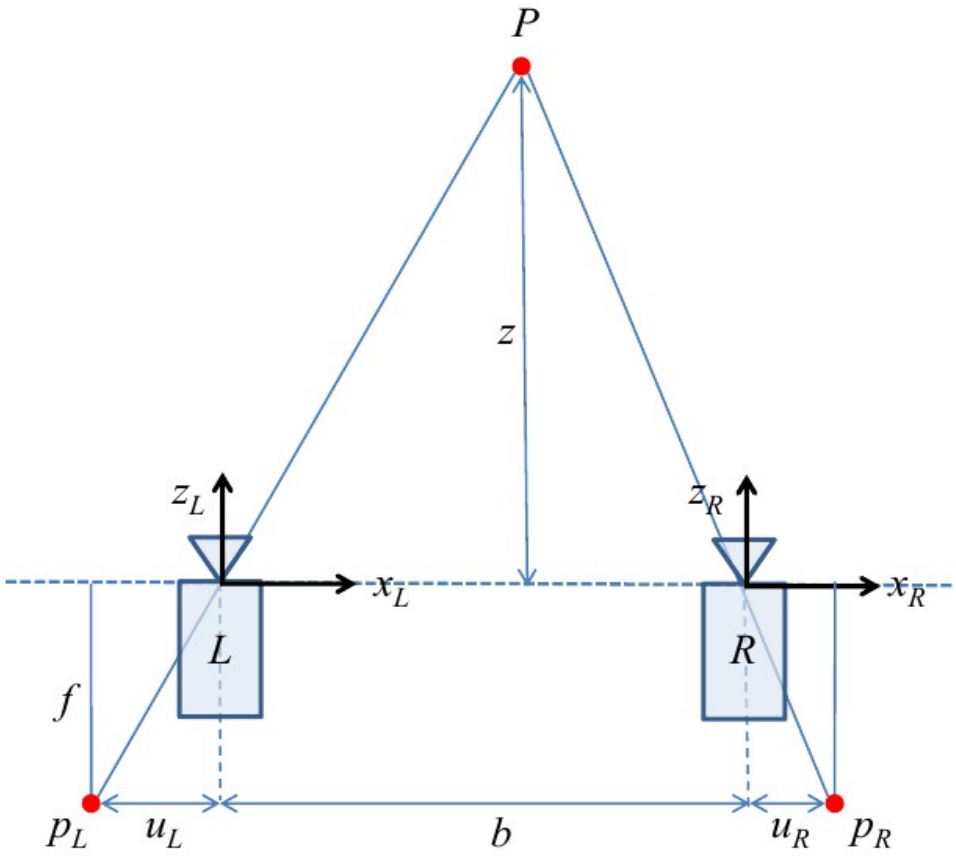


Figura AI.6: ejemplo de visualización de puntos de sistema estereoscópico

Fujifilm FinePix Real 3D W

La serie Fujifilm FinePix Real 3D W es una línea de cámaras digitales de nivel de consumidor. Diseñada para capturar imágenes estereoscópicas que recrean la percepción de la profundidad de 3D, tiene formatos de foto y vídeo 3D, sin perder modos de fotografía y vídeo estándar 2D. Estas disponen de un par de lentes (compensadas de izquierda a derecha por una línea de base que se aproxima a la distancia entre un par promedio de los ojos humanos), y una pantalla autoestereoscópica que compensan las imágenes para el ojo izquierdo y derecho de forma simultánea. La arquitectura de doble lente también permite nuevos modos de captura zoom

simultánea de una imagen 2D. El resto de la cámara es similar a otras cámaras digitales compactas.

El W1 tiene dos lentes, capturando imágenes de color con una resolución de 10 Megapíxeles y cada uno tiene zoom óptico 3x (35mm - 105mm en la cámara de 35mm equivalente). La pantalla a color LCD en la parte trasera de la cámara mide 2,8 "de diagonal, con 0,23 Megapíxeles. Puede ser electrónicamente cambiada entre pantalla normal y pantalla autoestereoscópica. Las dos lentes se pueden utilizar para tomar dos disparos simultáneos de la misma escena con diferentes configuraciones (zoom, ISO, etc.).

En agosto de 2010, Fujifilm anunció la W3, una nueva cámara estereoscópica compacta 3D con la capacidad de capturar imágenes y vídeos en 3D, siguiendo con características y especificaciones similares de la primera de su clase W1. El W3 cuenta con mayor resolución (720p) y un mejor rendimiento durante la noche, así como una pantalla autoestereoscópica mejor integrada.

Figura AI.7: cámara Fujifilm FinePix Real 3D W

4 Sistemas estereoscópico de codificación luz

La ecuación AI.9 se deriva de las relaciones relativas a los triángulos en la Figura AI.6. El hecho de que p_L y p_R en los sistemas de estéreo estándar se debe a la luz reflejada por P hacia las dos cámaras es secundario. El punto principal es la geometría del triángulo entre los rayos Pp_L, Pp_R y p_L, p_R. En esta perspectiva inspirada por un haz proyectado sobre una geometría desde un punto, en la que los puntos de imagen son equivalentes a los rayos que salen de dos centros de proyección. Cualquier dispositivo capaz de proyectar rayos entre su centro de proyección y los puntos de la escena puede considerarse funcionalmente equivalente a una cámara estándar . Este es el caso de los proyectores de luz, cualquier rayo de luz emitido por las conexiones de su centro de proyección al

punto de la escena P por la luz patrón de píxel p_A proyectada a P, como se muestra en la Figura AI.8.

En un sistema de visión estéreo donde una de las dos cámaras se sustituye por un proyector, es decir, compuesta por una cámara C y un proyector A como se muestra en la Figura AI.8, se denomina sistema de codificación activo o de luz. La cámara C tiene un Sistema de Coordenadas de la Cámara con coordenadas (x_C, y_C, z_C), también llamado sistema de referencia C-3D, y un sistema de referencia 2D con coordenadas (u_C, v_C) como se muestra en la Figura AI.8. El proyector similar tiene un Sistema de Coordenada del Proyector con coordenadas (x_A, y_A, z_A), llamada sistema de referencia A-3D, y un sistema de referencia en 2D con coordenadas (u_A, v_A). Como en el caso de sistemas pasivos estéreo estándar, los sistemas activos del tipo mostrado en la Figura AI.8 se pueden calibrar y rectificar con el fin de simplificar el proceso de estimación de profundidad.

La figura AI.8 muestra la proyección del patrón de luz píxel p_A, con coordenadas $p_A=[u_A,v_A]^T$ en el sistema de referencia A-2D. Para punto P de la escena 3D con coordenadas $P = [x,y,z]^T$ en el sistema de referencia C-3D. Si P no se ocluye, la proyección de luz radiada emitida por el proyector al píxel p_C de la cámara C establece el triángulo p_CPp_A. Si el sistema activo está calibrado y rectificado, p_C tiene coordenadas $p_C = [u_C=u_A+d, v_C=v_A]$. Como en el caso de los sistemas de estéreo estándar, ya que p_A y p_C son puntos conjugados, una vez que se conocen sus coordenadas la profundidad z de P, se puede calcular a partir de la ecuación AI.9, que en este caso se denomina triangulación activa desde A, es un sistema activo, y las coordenadas 3D de P se pueden calcular a partir de la ecuación AI.10 como se explicó en los sistemas de visión estereoscópica. La eficacia de los sistemas activos con respecto al problema de correspondencia puede ser fácilmente apreciable, cuando consideramos una pared recta de color uniforme. En este caso el píxel del patrón de luz p_A es proyectado sobre el punto P de la escena. De esta manera el píxel p_C de la cámara C donde se proyecta P (obviamente, en presencia de una pared recta no hay oclusiones) recibe de P el "color" de p_A y se hace reconocible entre sus píxeles vecinos. En este caso, los puntos conjugados p_A y p_C existe y puede también ser fácilmente detectado, mediante la adopción de patrones de luz adecuados.

Los sistemas de codificación de luz pueden proporcionar información en profundidad, también en caso de escenas sin características geométricas y de color donde los sistemas de estéreo estándar fallan al generar datos de profundidad. Las características de los patrones proyectados son fundamentales para la solución del problema, de correspondencia y el rendimiento general del sistema y su estudio, atrajo mucha atención. La proyección de secuencias de varias decenas de patrones de luz era típico de los métodos tempranos de codificación de luz, esto limita el rango de distancias medibles y su uso todavía en escenas. Este tipo de sistemas han sido y seguirá siendo la opción común para el modelado 3D de escenas fijas.

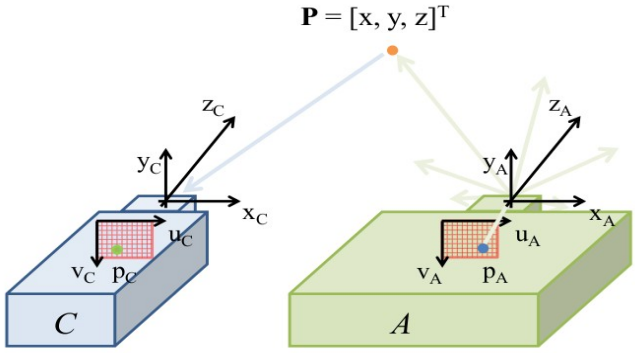


Figura AI.8: ejemplo de visualización de puntos de sistema estereoscópico con codificación de luz

Kinect™

La cámara Microsoft Kinect™ es una cámara de gama ligera, con patrón de luz, capaz de estimar la geometría 3D, de la escena adquirida a 30 fps y módulo VGA (640 × 480) resolución espacial. Su cámara con patrón de luz, la Kinect®, también tiene una de cámara de vídeo a color y un conjunto de micrófonos.

Desde el punto de vista funcional, la cámara Kinect® está basada en el chip PrimesensorTM,que es producido por Primesense, que está apoyada por una cámara de vídeo de infrarrojos y un un patrón de luz infrarrojos, para obtener una aplicación matricial del principio de triangulación activa y la estimación geométrica de escenas 3D. Además dispone de otra cámara RGB adicional para capturar los colores de la escena.

La cámara tiene un rango de trabajo entre 1.2 y 3.5 metros de distancia desde la posición de la cámara, puede llegar a 6m. La cámara posee un ángulo de visión de 57º horizontalmente y 43º verticalmente. Para funcionar en PC, Kinect® necesita alimentación externa a la que proporciona el puerto USB 2.0, por una toma de corriente.

Figura AI.9: cámara Kinect

Anexo II

Para entender el desarrollo matemático de las diferentes clases que se utilizan en la PCL se describirán los principios por los que se rigen las principales clases que se utilizaran en el proyecto. No es objeto este anexo de un estudio profundo del comportamiento sobre las nubes de puntos, ni el desarrollo de un análisis de su funcionamiento real.

1 Filtros

1.1 Filtros de eliminación de datos

Cuando la cantidad de datos que se obtienen a partir de un sensor son muy elevados, existe la posibilidad de reducir el número de elementos sin perder información, se realiza mediante mecanismos de poda de datos, disminuyendo la resolución de la imagen, donde existe la posibilidad de eliminar datos atípicos que pueden interferir en la calidad de la imagen.

En una fotografía digital vienen ordenados en forma de matriz, en cada celda está asignado un valor, que indica la intensidad de color. Dependiendo del volumen de datos que contenga la matriz de salida del sensor, viene definida la resolución. Por tanto, se obtiene mejores resoluciones para capacidades más altas. Con algunas excepciones ocurre de forma similar con los sensores de profundidad.

Se define la resolución de la nube, como la capacidad de obtener un número de puntos N de una sola captura sobre la escena. Para disminuir la resolución de la nube de puntos, tenemos que reducir el conjunto N_1 de puntos de la nube, a otra nube de puntos N_2 donde $N_1 > N_2$, los puntos de N_2 no tienen porque estar contenidos en N_1.

La disminución del tamaño de la nubes de puntos se realiza mediante técnicas heurísticas, se tiene en cuenta los datos circundantes de las nubes y cómo se ordenan en el formato, ya que pueden estar en forma de matriz con un criterio de orden, o establecer los puntos en forma de vector. En último caso, la nube de puntos se denomina que está desordenada. Para realizar la reducción de

puntos existen los siguientes métodos:

1.1.1 Tamizado en cubos 3D

El procedimiento de tamizado de una nube, con un número de puntos N_1, se realiza estableciendo el tamaños de los cubos (l_x,l_y,l_z), existiría un número n_1 de puntos. Se eliminan los puntos hasta dejar uno como representativo del cubo, obteniendo en cada cubo un conjunto resultante de n_2=1, solo si se cumple la condición previa en la cual $n_1 \geq 1$, en el caso de que n_1=0 el cubo no será representado en la nube de salida.

Cuando se aplican las transformaciones existe la posibilidad que no se produzca una reducción de puntos. Sino podemos asegurar que $N_2 \leq N_1$. El método no elimina datos atípicos que tengan dimensiones de profundidad. Si el punto no existe en el espacio, será evidente la reducción del número de puntos, como los datos con valor NaN. Los puntos de salida representarán el centroide del cubo, sobre el cual se ha realizado la eliminación de puntos, sin tener en cuenta la distribución de los puntos dentro del cubo.

1.1.2 Tamizado en cubos sobre el centroide

En este algoritmo se emplea la técnica de tamizados en cubos 3D, donde se establece los cubos sobre los que se aplicará la reducción de una nube N_1, con n_1 números de puntos. El tamaño de cubos con dimensiones (l_x,l_y,l_z), estarán distribuidos uniformemente sobre toda la nube de puntos. Y debe cumplirse que n_1>1 para representar el cubo, a través de un punto en la nueva nube de puntos N_2.

En este algoritmo, los puntos de la N_2, representa el centroide de los puntos que están dentro del cubo de tamizado, y no el centroide del cubo, por lo que el nuevo punto debe cumplir:

$$p_{i_2} = \frac{\sum_{j=1}^{n_1} \vec{p}_{j_1}}{n_1} \qquad \text{(AII.1)}$$

1.1.3 Reducción del tamaño matriciales

Las nubes de puntos pueden organizarse en forma matricial, el motivo es facilitar algunos métodos que han sido heredados de las imágenes digitales. En estas se realiza la reducción de resolución, eliminado cierto número de filas y columnas.

Para aplicar este criterio, es necesario tener una nube de puntos que esté organizada en filas y columnas n_1xm_1, una condición indispensable. Conociendo el tamaño de la matriz que se tiene que reducir se puede aplicar el filtro de las siguientes formas:

1. Reducción débil: se eliminan las filas y columnas múltiplos de r, siendo r un número entero. En este caso la matriz mantienen un número mayor de la mitad de los datos.

2. Reducción 1:1; se borran las filas y columnas pares o impares y de mantienen las inversas reduciendo la matriz con la mitad de datos

3. Reducción fuerte; se mantienen las filas y columnas múltiplos de un entero r, en este caso la matriz se reduce manteniendo menos de la mitad de los datos iniciales.

Este tipo de filtros son adecuados para los sensores que generan nubes ordenadas con disposición matricial, se debe considerar que no tienen en cuenta el ratio de puntos volumétricos, no se controla la información que se pierde.

1.1.4 Reducción de datos atípicos

La obtención de datos a través de un sensor implica que el ruido afecte al resultado. Por este motivo se producen, según la calidad del sensor, distorsiones. Para evitar contener información errónea, durante la etapa de procesamiento, es necesario el estudio de los que se consideran puntos atípicos, para proceder a su eliminación. No solo conseguimos una reducción de la resolución de la imagen, sino que el ruido no afecte al resultado final en las siguientes etapas.

Cuando un dato es afectado por el ruido, se produce lo que denominan puntos atípicos, estimados en función del número de vecinos circundantes, estableciendo un umbral h no muy pequeño. Si no existe ningún punto vecino de la misma nube se considera que es un valor atípico y se procede a su eliminación. El procedimiento del algoritmo se realiza de la siguiente forma:

1. Establecemos el umbral h.

2. Buscamos los puntos que se encuentran a una distancias $d<h$.

3. Si el número de puntos cercanos $n_i > 0$ pasa a formar parte de la nueva nubes de puntos N_2.

1.2 Filtros de corrección de datos

En muchos sensores los datos obtenidos son perturbados por diversas fuentes de ruido, que distorsiona la imagen. Entregando datos que se encuentran en alguna medida distorsionados. También existe la posibilidad de tener sensores en los que capturan las nubes de una forma en que

se establecen errores de medida. Surge la necesidad de buscar filtros que corrijan los valores, obteniendo así los que son más fieles a la realidad capturada.

El objetivo de este tipo de filtros no es realizar una reducción del tamaño de los datos, sino el manteniento de todos y corregir sus valores en función de los circundantes. Las técnicas establecidas son heredadas de la visión artificial en 2D o procesamiento de imágenes 2D, ajustando el tipo de filtro para el nuevo tipo de datos a corregir.

1.2.1 Filtrado bilateral subyacente

El enfoque de este método es corregir los datos teniendo en cuenta la vecindad más cercanas mediante un filtro paso bajo. Puede tener problemas cuando se trata de corregir datos en un borde donde se produce un cambio muy fuerte.

Si aplicamos el filtro paso bajo sobre una imagen **f**(x) obtenemos la siguiente salida:

$$\boldsymbol{h}(\boldsymbol{x}) = k_d^{-1}(\boldsymbol{x}) \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \boldsymbol{f}(\xi)\, c(\xi, \boldsymbol{x})\, d\xi \qquad \text{(AII.2)}$$

donde c(ξ,**x**) mide la cercanía geométrica entre el centro del barrio **x** y un punto cercano ξ. El tipo de letra negrita para **f** y **h** hace hincapié en el hecho de que tanto las imágenes de entrada y salida pueden ser multibanda. Si el filtrado de paso bajo es preservar la componente continua de las señales de paso bajo obtenemos

$$k_d(\boldsymbol{x}) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} c(\xi, \boldsymbol{x})\, d\xi \qquad \text{(AII.3)}$$

Si el filtro es invariante c(ξ,**x**) es solo una función de la diferencia vectorial ξ-**x,** y k_d es constante. El rango de filtrado se define de manera similar:

$$\boldsymbol{h}(\boldsymbol{x}) = k_d^{-1}(\boldsymbol{x}) \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \boldsymbol{f}(\xi)\, s(\boldsymbol{f}(\xi), \boldsymbol{f}(\boldsymbol{x}))\, d\xi \qquad \text{(AII.4)}$$

excepto, que ahora s(**f**(ξ), **f**(x)) mide la similitud de parámetros entre el píxel en el centro de barrio **x** y la de un punto cercano ξ. La función de similitud s opera en el rango de la función de imagen **f**, mientras que la función de la cercanía c opera en el dominio de f. La constante de normalización k_d(**x**)se sustituye por:

$$k_r(\boldsymbol{x}) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} s(\boldsymbol{f}(\xi), \boldsymbol{f}(\boldsymbol{x}))\, d\xi \qquad \text{(AII.5)}$$

en contraposición a lo que ocurre con la función de la cercanía c, la normalización de la función de similitud s, depende de la imagen **f**. Decimos que la función de similitud s es imparcial si solo depende de la diferencia **f**(ξ)- **f**(x).

La combinación del dominio y filtrado de rango se denotará como el filtrado bilateral.

Reemplaza t el valor del píxel en **x,** con un promedio de los valores de píxeles similares y cercanos. En las regiones lisas, los valores de píxeles en un pequeño barrio son similares entre sí. La función de similitud normalizado k^{-1} es cercano a uno. Como consecuencia, el filtro bilateral actúa esencialmente como un filtro de dominio estándar sobre los promedios de diferencias de distancias pequeñas, débilmente correlacionados entre valores de píxeles causadas por el ruido.

1.2.2 Convolución

La convolución es un operador matemático que transforma dos funciones **f** y **g** en una tercera función, que en cierto sentido, representa la magnitud en la que se superponen **f** y una versión trasladad e invertida de **g**. Una convolución es un tipo muy general de media móvil.

La convolución de **f** y **g** se denota **f*g**. Se define como la integral del producto de ambas funciones después de desplazar una de ellas una distancia t.

$$(f*g)(t)=\int_{-\infty}^{\infty} f(\eta)\,g(t-\eta)\,d\eta \qquad \text{(AII.6)}$$

El intervalo de integración dependerá del dominio sobre el que estén definidas las funciones. En el caso de un rango de integración finito, ***f*** y ***g*** se consideran a menudo como extendidas, periódicamente en ambas direcciones, de forma que el término ***g*(*t* - η)** no implique una violación en el rango. Cuando usamos estos dominios periódicos la convolución a veces se llama cíclica. También es posible extender con ceros los dominios. El nombre usado cuando ponemos en juego estos dominios "cero-extendidos" o bien los infinitos es el de **convolución lineal**, especialmente en el caso discreto que presentaremos en la expresión AII.7.

Si **x** e **y** son dos variables aleatorias independientes con funciones de densidad de probabilidad ***f*** y ***g***, respectivamente, entonces la densidad de probabilidad de la suma ***X* + *Y*** vendrá dada por la convolución ***f* * *g***. Para las funciones discretas se puede usar una forma discreta de la convolución. Esto es:

$$f[m]*g[m]=\sum_{n} f[n]\,g[m-n] \qquad \text{(AII.7)}$$

Cuando multiplicamos dos polinomios, los coeficientes del producto están dados por la convolución de las sucesiones originales de coeficientes, en el sentido dado aquí (usando extensiones con ceros como hemos mencionado).

Generalizando los casos anteriores, la convolución puede ser definida para dos funciones de cuadrado integrable definidas sobre un grupo topológico, localmente compacto. Una generalización diferente es la convolución de distribuciones.

1.2.3 Núcleo gaussiano

El núcleo gaussiano se define matemáticamente como:

$$G_{ND}(\vec{x},\sigma)=\frac{1}{\left(\sqrt{2\pi}\sigma\right)^{N}}\mathrm{e}^{\frac{-|x^2|}{2\sigma^2}} \quad \text{(AII.8)}$$

donde σ determina el ancho del núcleo. Esta función estadística, es aplicada sobre los parámetros de la nube, produciendo un alisamiento en el parámetro modificado.

1.2.4 Mediana

Para filtrar los errores es necesario cuantificar las posibles desviaciones con la vecindad. Estos procesos de estimación y transformación pueden requerir mucho tiempo de cómputo. En el caso de la mediana, que es un parámetro estadístico donde se trata de encontrar entorno a los puntos vecinos el valor central, transformando el antiguo valor por la mediana del conjunto de sus vecinos.

Con este tipo de filtro se consigue resaltar los cambios producidos en la imagen, con un método sencillo que no requiere procesos de iteración.

2 Keypoints

Los detectores de keypoint o puntos de interés se clasifican principalmente, según el tipo de datos que se dispongan. Cada uno de ellos utiliza diferentes bases matemáticas para la obtención de los puntos. En este apartado, se desarrolla la base matemática de cada uno de los detectores.

2.1 Detectores de escala fija

Los detectores de escala fija encuentran los puntos claves a una escala constante, para esto hay que proveer al algoritmo de parámetro con una métrica adecuada. Como se explica en la figura AII.1 estos se pueden resumir en dos pasos:

Un primer paso inicial, opcional, es la poda de datos de entrada por umbralización. Una parte para reforzar la selección de los puntos más significativos, por un criterio adicional con respecto a las principales mediciones destacadas, desplegado en las etapas subsiguientes. Además de mejorar la eficiencia del algoritmo, mediante la reducción del número de puntos previstos, como entrada para los pasos siguientes.

El segundo paso, consiste en un procedimiento de Supresión de Falsos Máximos, sobre una medida calculada en cada punto, no descartados en la poda inicial. La medición asociada puede ser

de un punto en juicio (una propiedad de un vértice de la malla) o regiones en juicio (una propiedad de una región alrededor de cada vértice). En el caso de puntos destacados en juicios, la escala de entrada es usada para definir el tamaño de soporte de los falsos máximos. Con regiones destacadas en juicio, la escala define el soporte sobre las mediciones de prominencia. El soporte de falsos máximos se define mediante un parámetro adicional.

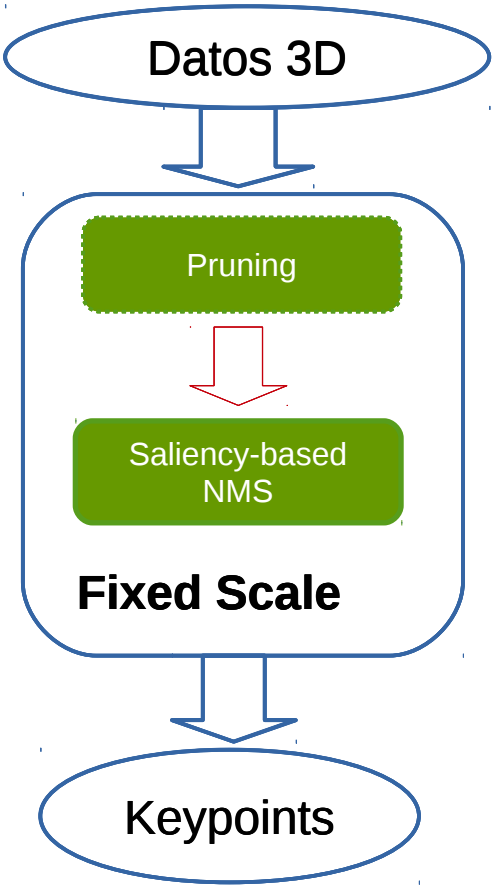


Figura AII.1: método de extracción de puntos de interés para escala fija

2.1.1 Harris 3D

El método Harris propuesto por Harris and Stephens en 1998, para la detección de puntos de interés en imágenes 2D, es una técnica muy popular debido a su fuerte invarianza a la rotación, la escala, la variación de la iluminación y ruido de la imagen. El detector de Harris se basa en la función de autocorrelación local de una señal. Mide los cambios locales de la señal con parches desplazado por una pequeña cantidad en diferentes direcciones. La autocorrelación local se define como:

$$e(x,y)=\sum_{x_i,y_i} W(x_i,y_i)\left[I(x_i+\Delta x, y_i+Vy)-I(x_i,y_i)\right]^2 \qquad \text{(AII.9)}$$

donde I (·, ·) denota la función de imagen y (x_i, y_i) son los puntos en la función gaussiana W sobre

(x, y), que define la zona de vecindad en el análisis.

Si usamos el desarrollo de Taylor aproximado, truncado en los primeros términos de orden sobre la imagen, obtenemos:

$$e(x,y)=S\begin{bmatrix}\sum W.I_x^2 & \sum W.I_x.I_y \\ \sum W.I_x.I_y & \sum W.I_y^2\end{bmatrix}S^T=S\,E(x,y)\,S^T \qquad \text{(AII.10)}$$

donde S=[Δx Δy] es un vector de cambio, I_x y I_y denota la derivadas parciales en x e y, junto con W es evaluado en el punto (x_i,y_i) y E(x,y) es denominada la matriz de covarianza.

Harris y Stephens propusieron para analizar los valores propios de la matriz E, que contiene suficiente información local relacionada con la estructura de vecindad. Para evitar el complicado cálculo de valores propios, se propone asignar a cada pixel en la imagen el siguiente valor:

$$h(x,y)=det(E)-k(tr(E))^2 \qquad \text{(AII.11)}$$

donde k es constante.

El operador de Harris se ha utilizado en muchas aplicaciones en el procesamiento de imágenes y visión por computador por su simplicidad y eficiencia. Sin embargo, el problema con los datos 3D es que la topología es arbitraria y no está claro cómo calcular los derivados. Para hacer frente a este problema, con nubes de puntos, se analiza una característica geométrica local, como es la normal, por lo que pasamos de la siguiente manera a determinar la matriz de covarianza:

$$E=\sum_{i\in W} D_i\cdot D_i^T \rightarrow E\,3D=\sum_{i\in W} N_i\cdot N_i^T \qquad \text{(AII.12)}$$

Cuando introducimos la característica geométrica local de las normales, no tenemos en cuenta las posibles esquinas que se podrían detectar por el efecto de la intensidad del brillo, cómo serían las esquinas dibujadas sobre planos. Por tanto, algunos autores decidieron agrupar ambos conceptos para detectar ambos tipos de esquinas, teniendo un espacio con 5 grados de libertad, aunque utilizamos un espacio de 6 dimensiones, los puntos debido a la escala de grises están sobre un plano, este tipo de detector se les denomina harris6D. Para ello se crea un vector de normal ampliable con gradiente de intensidad, de tal manera:

$$G_i=\left(N_i^T,D_i^T\right)^T=\left(N_x,N_y,N_z,D_x,D_y,D_z\right) \qquad \text{(AII.13)}$$

La matriz de covarianza se determinaría con el siguiente operador:

$$E\,6D=\sum_{i\in W} G_i\cdot G_i^T \qquad \text{(AII.14)}$$

El método propuesto por Harris y Stephens ha funcionado con unas buenas características, pero algunos desarrolladores se ha afanado en mejorarlos, utilizando el método similar con la matriz de covarianza, para detectar los keypoints, a través de las nubes de puntos. Los métodos que se han generalizado para este problema se explican:

$$\text{Harris} \quad h(x,y,z) = det(C) - k(tr(E))^2 \tag{AII.15}$$

$$\text{Noble} \quad h(x,y,z) = \frac{det(E)}{tr(E)} \tag{AII.16}$$

$$\text{Lowe} \quad h(x,y,z) = \frac{det(E)}{(tr(E))^2} \tag{AII.17}$$

$$\text{Tomasi} \quad h(x,y,z) = \lambda_{min} \tag{AII.18}$$

Al igual que en las imágenes 2D, se estudia el valor de la respuesta aplicando un umbral mínimo que deben tener para que el detector lo selecciones como keypoint. Además es necesario aplicar un radio para evitar vecindades muy cercanas entre ellos.

2.1.2 SUSAN

El detector de esquinas de SUSAN fue desarrollado por S. M. Smith y J. M. Brady [5], este lo que hace es emplear un enfoque morfológico. Se trata de una técnica de procesamiento de imágenes de bajo nivel. Además ha sido utilizada en la detección de bordes y la eliminación de ruido.

Este detector funciona de la siguiente manera: Para cada píxel de la imagen consideramos los vecinos en el interior de un círculo de radio fijo alrededor de él. El píxel central es referido como el núcleo y su valor de intensidad es tomado como referencia. Todos los píxeles en el interior de ese círculo son clasificados en dos grupos: los de intensidad parecida al núcleo o los que poseen un valor de diferente. De este modo, cada punto de la imagen tiene asociada un área local de brillo similar, su tamaño está asociado a información importante sobre la estructura de la imagen en dicho punto.

En aquellas partes de la imagen homogéneas, el área local de brillo similar cubre casi la totalidad del círculo de vecindad. Al encontrarnos en presencia de un borde, el área cubre alrededor del 50 % del círculo y en presencia de una esquina este porcentaje cae por debajo del 25 % (ver figura AII.2. Por tanto, las esquinas pueden ser localizadas como los puntos de la imagen, en los que el número de píxeles de intensidad similar en el círculo de vecindad es mínimo y está por

debajo de un cierto umbral predefinido. Para darle mayor robustez al método, los píxeles más próximos al núcleo reciben un mayor valor de ponderación. Se declaran una serie de reglas para eliminar aquellos descriptores que no interesan. Finalmente, los mínimos locales son los seleccionados como candidatos.

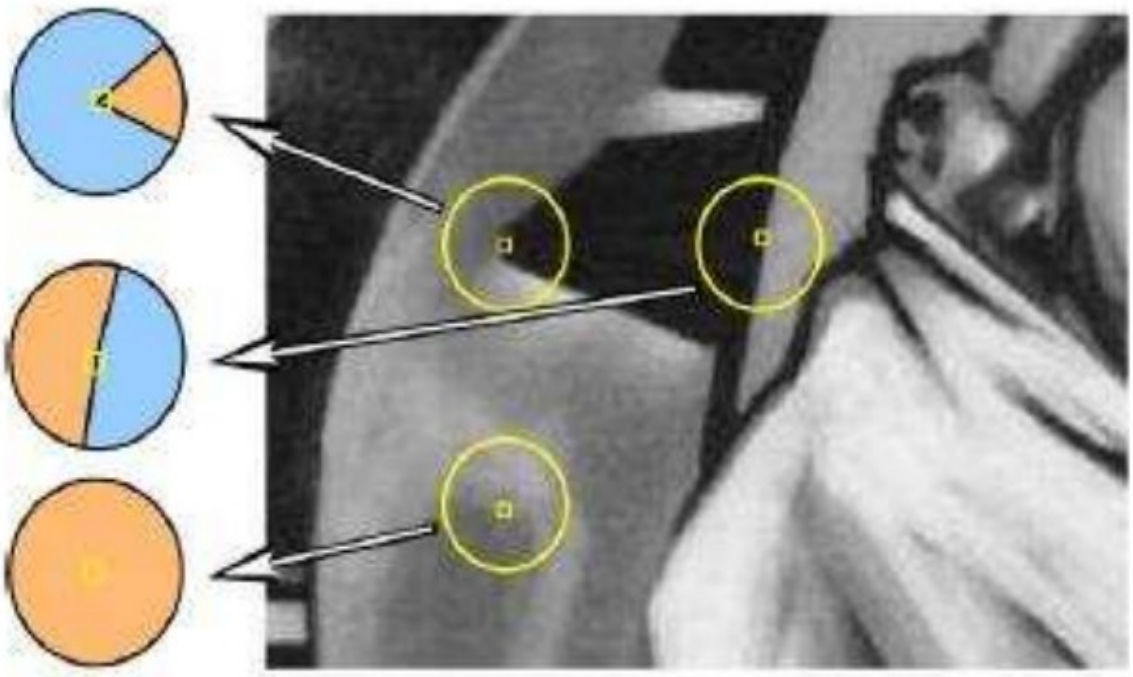


Figura AII.2 : detector de SUSAN. Distinción entre un área homogénea, un bordeo una esquina.

Inicialmente estaba preparado para imagen 2D(sería de esperar que este algoritmo fuese con cambio de escala), aunque está basado en un nuevo enfoque del método SUSAN, que también describe cómo ejecutarlo en 2D. Este algoritmo está preparado para escala fija con imagen 3D. En este nuevo enfoque trata de reconocer los puntos claves a través del estudio de un píxel o punto que se denomina núcleo, con el enmascaramiento de los puntos más cercanos. Analizando uno de los parámetros que puede ser la componente normal. En este algoritmo se ejecutará la siguiente función con los vecinos más cercanos o área USAN:

$$c\left(\vec{r},\vec{r_0}\right)=e^{-\left(\frac{I_{\vec{r}}-I_{\vec{r}_0}}{t}\right)^6} \qquad \text{(AII.19)}$$

donde $\vec{r_0}$ es el núcleo, el punto que vamos a comparar con respecto a los más cercanos, serán definidos por $\vec{r}$ y el parámetro t, establece el umbral que definiremos. No podemos hacer una comparación píxel a píxel con esta fórmula. Si queremos establecer qué puntos son de interés y

cuáles no, debemos aplicar la suma de la comparación con todos los vecinos, de la siguiente manera:

$$n(\vec{r}_0) = \sum_{\vec{r}} c(\vec{r}, \vec{r}_0) \tag{AII.20}$$

Ahora con $n(\vec{r}_0)$ podemos estudiar los diferentes valores con los que intentaremos hallar los máximo locales.

Una de las características principales del método es estar preparado inicialmente para detectar bordes, podemos esperar que existan puntos de interés alineados. Esto hace que la información que transmite los puntos de interés no sea suficientemente clara.

2.1.3 Firma intrínseca de forma (ISS)

Una *firma intrínseca de forma* consiste en un marco de referencia que permite tanto la extracción de características, desde un punto de vista invariante y un rápido registro. Además tiene un vector de características altamente discriminatorio, que codifica la forma 3D. Para proponerlo, como método de extracción de puntos claves, es necesario obtener previamente la firma para cada punto.

En descriptores de forma es deseable analizar desde un punto de vista independiente la forma local del objeto. La invarianza se logra mediante a un punto de vista independiente. El vector normal al calculado a través del parche, que forma la vecindad, ha sido la elección unánime para construir descriptores de regiones independientes a la vista. Aunque es popular el vector normal a la superficie por sí solo no es suficiente para definir un sistema de coordenada 3D.

Se define un marco de referencia intrínseco F_i, en un punto base p_i, con un soporte de marco con radio $r_{density}$ usando el análisis de los autovalores de la matriz de dispersión como sigue:

1. Cálculo del peso en cada punto, para ello se impone el radio de esfera y el peso es inversamente proporcional al número de puntos que hay dentro de la esfera $r_{density}$, con los puntos vecinos, de tal manera:

$$w_i = \frac{1}{\left\| \left(p_j : |p_j - p_i| \leq r_{density} \right) \right\|} \tag{AII.21}$$

este peso es usado para compensar las desigualdades de muestreo, en los puntos 3D, por lo que las regiones de muestreo con mayor separación contribuyen más que las de mayor densidad.

2. Calcula una matriz de dispersión usando todos los puntos p_j con distancia < r_{frame} como se ve en la fórmula:

$$COV(p_i) = \frac{\left(\sum_{|p_j - p_i| \le r_{frame}} w_j (p_j - p_i)(p_j - p_i)^T \right)}{\sum_{|p_j - p_i| \le r_{frame}} w_j} \quad \text{(AII.22)}$$

3. Se obtienen los autovalores de la matriz y se ordenan de forma decreciente para obtener los autovectores ordenados.

4. Usa como origen el punto de estudio para los autovectores y se calcula el eje -z por ortogonalización de autovectores.

El detector utiliza la magnitud de los valores propios más pequeños (para incluir solo puntos con grandes variaciones a lo largo de cada dirección principal) y la relación entre dos valores propios sucesivos (para excluir puntos que tienen propagación similar a lo largo de las direcciones principales).

La firma intrínseca de forma $S_i = \{F_i, f_i\}$ de un punto p_i consiste en un marco de referencia intrínseco $F_i=\{p_i,\{e_i^x,e_i^y,e_i^z\}\}$, donde p_i es el origen y $\{e_i^x,e_i^y,e_i^z\}$ es el conjunto de vectores de la base.

El marco de referencia de las características intrínseca de forma del objeto es independiente al punto de vista. Por lo tanto, estas son independiente a la visión y puede ser calculado usando el marco como referencia.

Sin embargo, su base $\{e_i^x, e_i^y,e_i^z\}$, que especifica los vectores de sus ejes con respecto al sistema de coordenadas del sensor, depende del punto de vista y codificado directamente, transformado el sistema del sensor a la orientación del marco intrínseco local. Esto permite coordinar rápidamente la referencia calculada y la vista de registro.

2.1.4 Características NARF (Normal Aligned Radial Feature)

Dos de los métodos más populares para la extracción de puntos de interés y la creación de descriptores estables en el área de visión por computador 2D son SIFT y SURF. Estos métodos se basan en gradientes locales y una orientación única, en una región de la imagen para lograr la invariancia a la rotación. Aunque SIFT y SURF no se pueden trasladar directamente al espacio 3D, muchos de sus conceptos son útiles.

Las características NARF [16] proporcionan un método de extracción de puntos así como un descriptor de puntos 3D. Estas se generan en puntos donde la superficie es estable pero cambia de forma significativa en el entorno de vecindad más cercano. Además, hace un uso explícito de la información de bordes.

Un aspecto importante de la extracción de características es el tratamiento de los bordes. Estos suelen aparecer como líneas no continuas de primer plano, al fondo. Existen tres tipos de bordes relevantes para este algoritmo (ver Figura AII.3):

1. Bordes de objetos (obstacle border): Puntos visibles más exteriores que pertenecen al objeto.

2. Bordes de sombras (shadow borders): Puntos en el fondo contiguos a oclusiones.

3. Puntos velo (veil points): Puntos interpolados entre los bordes de los objetos y los bordes de sombra. El correcto tratamiento de estos puntos mejora claramente los resultados posteriores de comparación y clasificación.

Existen diversos indicadores para la detección de bordes. En este caso, se utiliza un cambio en la distancia entre puntos vecinos. Este indicador es además muy robusto ante ruido y cambios de resolución. Para clasificar los bordes se siguen estos pasos:

1. Utilizar una heurística para encontrar la distancia 3D euclídea entre puntos vecinos que no pertenecen a un borde.

2. Utilizar esta información para calcular una puntuación que determine si dicho punto forma parte de un borde.

3. Identificar la clase del punto del borde.

4. Utilizar una supresión no-máxima para encontrar la posición exacta del borde. Es decir, descartar aquellos puntos por debajo del máximo local y de un umbral preestablecido (0.8).

El siguiente paso es detectar los puntos de interés. Para ello se establecen una serie de requisitos:

1. Debe utilizar información de los bordes y de la estructura de la superficie.

2. Deben seleccionarse puntos que se puedan detectar, incluso si el objetos se observa desde otra perspectiva.

3. Los puntos deben encontrarse en áreas estables que permitan la estimación de la normal y el cálculo del descriptor.

Los puntos de interés estables necesitan cambios de superficie considerables en una vecindad local para poder ser detectados de forma robusta independientemente de la perspectiva. Para esto, hay que:

1. Comprobar el entorno de vecindad de cada punto de la imagen y determinar una puntuación, según cuánto cambia la superficie en dicha posición y la dirección dominante en ese cambio. También se debe incorporar la información de los bordes.

2. Consultar las direcciones dominantes alrededor de cada punto y calcular un valor de interés que represente:

   a) Cuánto difieren esas direcciones entre sí.

   b) Cuánto cambia la superficie del punto.

3. Ajustar los valores de interés.

4. Realizar una supresión no-máxima para encontrar los puntos de interés finales.

El parámetro más relevante en este proceso es el tamaño de soporte (support size), que es el diámetro de la esfera alrededor del punto de interés,que incluye a todos los puntos del entorno de vecindad cuyas direcciones dominantes, se utilizan para el cálculo del valor de interés. Este mismo se utilizará posteriormente para determinar qué puntos se considerarán en el cálculo del descriptor.

En la Figura AII.3 se puede observar el procedimiento de extracción de puntos clave. Podemos observar una imagen de profundidad, con los bordes extraídos marcados(los distintos tipos de bordes aparecen marcados con diferentes colores). En b aparecen las puntuaciones de cambio de superficie según los bordes y el principio de curvatura. En c están señalados los valores de interés para un tamaño de soporte de 20 cm, mientras que en d están señalados dichos valores para un tamaño de soporte de 1m.

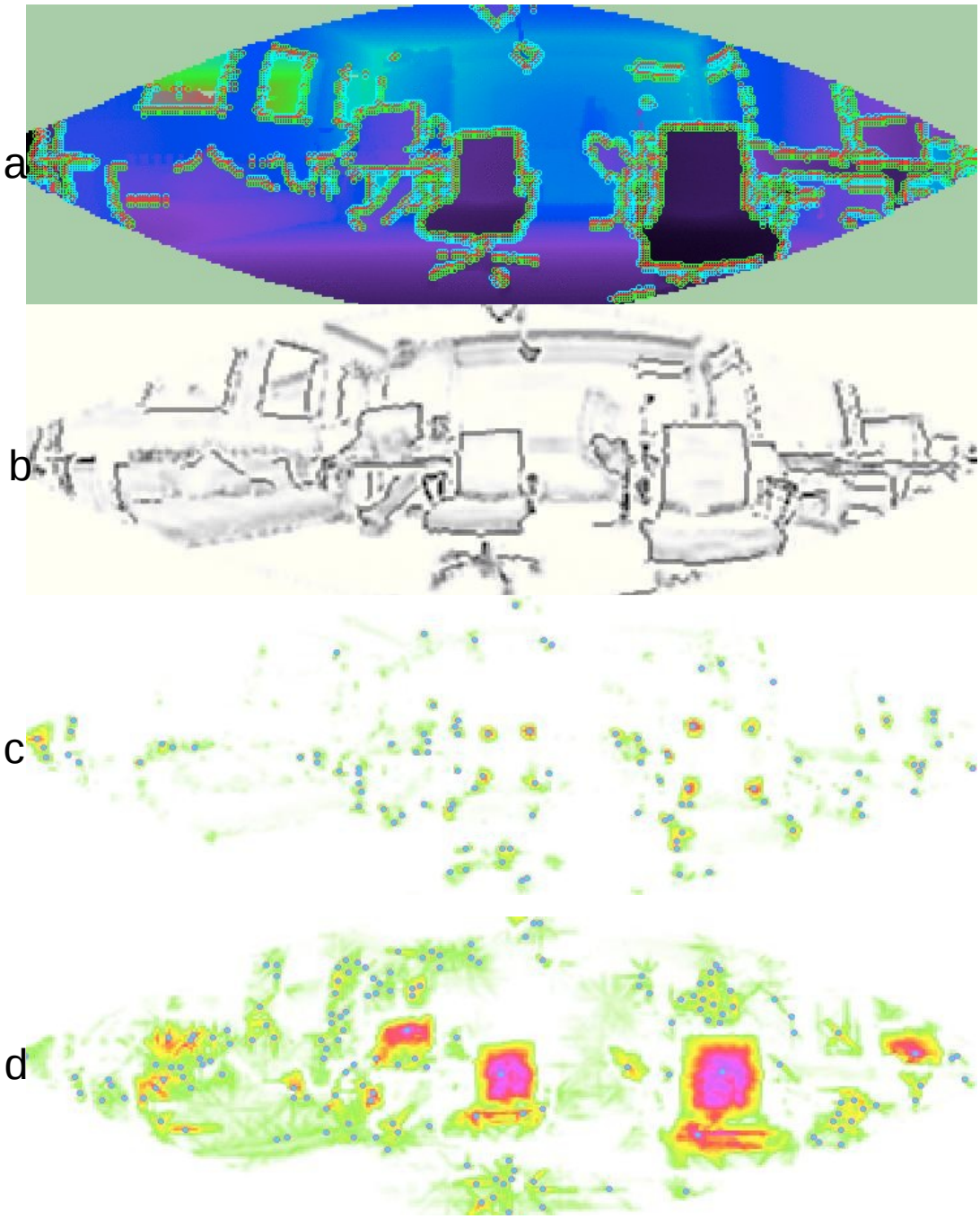


Figura AII.3: estudio Visual de características NARF para extracción de keypoints

El último paso del proceso es el cálculo del descriptor. Estos describen el área alrededor de un punto de interés de forma que su posterior comparación sea eficiente. Los NARF pretenden:

1. Capturar la existencia de espacio libre y ocupado, de forma que se puedan describir partes de la superficie y la forma exterior de los objetos.

2. Que sea robusto frente a ruido en el punto de interés.

3. Que permita extraer un único espacio de coordenadas local.

De forma similar a SIFT y SURF, el descriptor NARF permite obtener una orientación única respecto a la normal.

En definitiva, para generar el descriptor NARF se siguen los siguientes pasos:

1. Calcular un rango alineado a la normal en el punto, es decir, una pequeña imagen con el observador mirando directamente al punto a través de la normal.

2. Superponer un patrón en estrella, de forma que cada rayo corresponda a un valor del descriptor final. De esa forma se captura cómo cambian los píxeles bajo el rayo.

3. Extraer una orientación única.

4. Desplazar el descriptor según su valor para que sea invariante a rotación.

En la Figura AII.4 aparecen los pasos de generación del descriptor NARF para un punto. En este se puede observar la imagen de profundidad. Por ejemplo con un sillón al frente, la cruz negra señala el punto de interés. En b se puede visualizar cómo el descriptor es calculado. Se muestra una región de la esquina del sillón y debajo el descriptor, cada una de las celdas corresponde a cada uno de los rayos (verde) y la flecha roja señala la orientación dominante extraída.

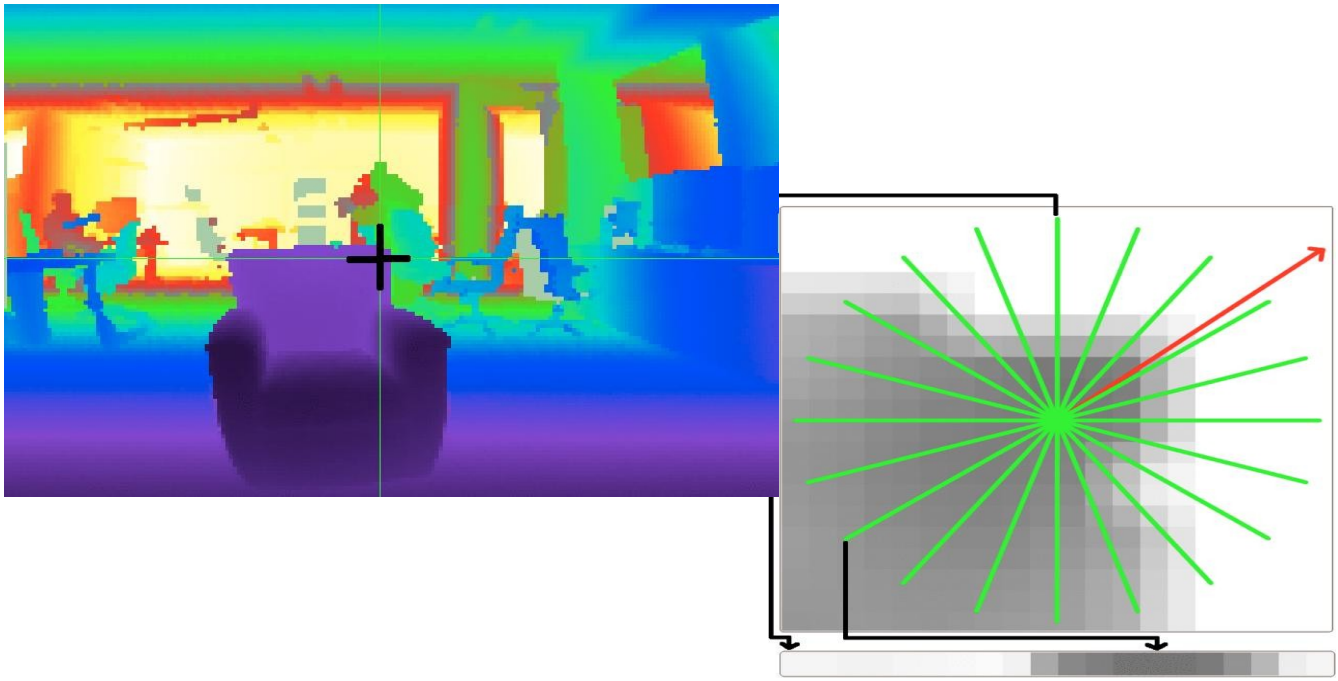


Figura AII.4: explicación gráfica de obtención de descriptor NARF

2.2 Detectores con adaptación de escala

La estructura común de los detectores adaptados a escala incluye la construcción de un espacio de escala definida en la superficie, extendiendo así directamente al caso de los datos 3D, del concepto conocido definido para imágenes 2D. Alternativamente, en lugar de extender la teoría de escala-espacio para datos 3D, se puede realizar una incrustación de los datos sobre un plano 2D, con el fin de llevar a cabo el análisis escala-espacio tradicional.Con todos los puntos en una escala asociada, se asocia un parámetro a una función adecuada a la escala, esta escala servirá de apoyo a procesos subsiguientes. A menudo, la presencia de máximos locales múltiples para puntos significativos con diferentes escalas, se detectan en la misma ubicación. Si la función muestra una tendencia monótona, entonces hay una escala característica que puede ser definida y el punto se descarta desde el conjunto de puntos claves candidatos.

Los puntos significativos son recogidos por medio de un método de supresión de falsos máximos en la escala característica de cada punto. Este método de la etapa de supresión de falsos máximos se lleva a cabo tanto espacial como a lo largo de la dimensión de la escala, aunque podría hacerse solo espacialmente. Utilizando la escala característica para definir el soporte sobre el qué se calcula los valores destacados.

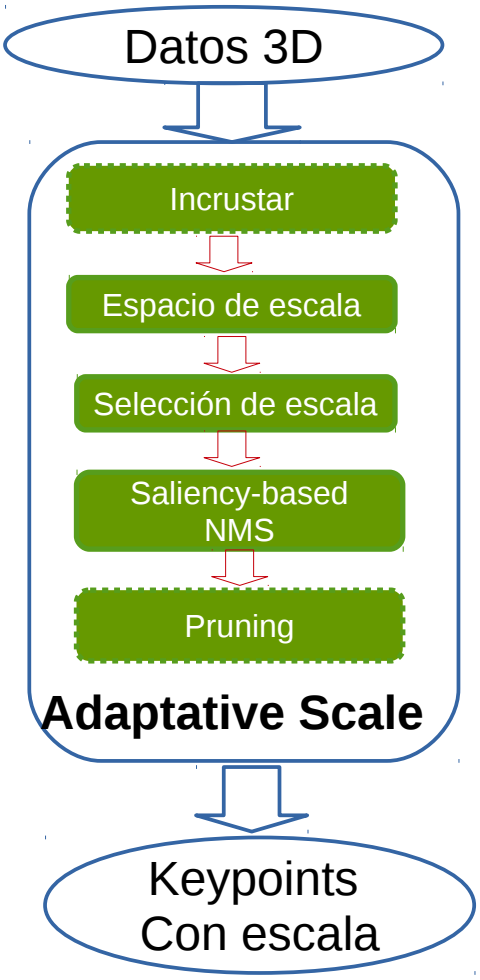


Figura AII.5: Flujo para la extracción de puntos de interés con variación de escala

Por último, se pueden ejecutar con fines similares a los de los métodos de escala fija, una etapa de poda opcional mediante los puntos adicionales que son eliminados en base a diferentes limitaciones.

2.2.1 AGAST

El detector de puntos de interés AGAST(*Adaptive and Generic Accelerated Segment Test*) utiliza técnicas heredadas de métodos desarrollados anteriormente denominados FAST(*Features from Accelerated Segment Test*) y SUSAN(*Smallest Uni-Value Segment Assimilating Nucleus Test*), de la detección de esquinas, para comprenderlo mejor describiremos el método FAST, como se ha adaptado a la configuración del algoritmo, para conseguir mayor eficiencia y resultados.

Revisión del método FAST

El principio de FAST se basa en el detector de esquinas SUSAN. Estos métodos utilizan un área circular para determinar los píxeles vecinos más brillantes y más oscuros. En el FAST, no se evalúa todo el área del círculo, sino el segmento discretizado de los píxeles que describen el círculo. En los dos métodos calculan el círculo utilizado de máscara, el algoritmo de Bresenham, como parámetro más utilizado se toman radios de 3 o 4 píxeles. Para un test acelerado el segmento de 16 píxeles tiene que ser comparado con el valor del núcleo. Para evitar que este test sea tan extenso, el criterio de esquina es más relajado.

Los criterios para que un píxel sea esquina o keypoint, de acuerdo con el test de segmento acelerado(AST) es el siguiente: debe haber al menos S píxeles conectados en el círculo que son más brillantes o más oscuros, que un umbral determinado por el valor del píxel central. Los valores de los otros 16-S píxeles no se tienen en cuenta. El valor de S define el ángulo máximo de la esquina detectada. Manteniendo S lo más grande posible, mientras que la supresión de bordes (donde S=8), aumenta la repetibilidad del detector de esquinas. De forma general, se toman tamaños de 9 segmentos (FAST-9). También se aplica un umbral de diferencia mínima (t) para comparar el valor del píxel en el segmento circular con el brillo del núcleo. Este parámetro controla la sensibilidad de la respuesta del vértice. En valores altos de t se obtienen esquinas con ángulos más fuertes, mientras que para valores pequeños las esquinas tienen ángulos más suaves.

Una cuestión sigue siendo, saber qué píxel se ha de comparar primero, segundo, tercero, y así sucesivamente. Obviamente, hay una diferencia en la velocidad,si un píxel consecutivo tras otro se evalúa. Por ejemplo: de bisección en el patrón de círculo se utiliza para probar si el criterio de esquina se aplica o no se puede aplicar más. Este tipo de problema se conoce como limitación por el paradigma veinte preguntas. Cuando pregunta cuál es el resultado de una pregunta en una decisión de árbol con el objetivo de elegir la longitud más corta. Se utilizan algoritmos de inteligencia artificial como ID3 con métodos de aprendizaje, para encontrar el mejor árbol basado en datos de entrenamiento del entorno donde FAST es aplicado. No se garantiza que todas las configuraciones

posibles de píxeles se encuentran, ya que pequeñas rotaciones de la cámara pueden producir configuraciones de píxeles que no han sido medidos en las imágenes de prueba. Si todas las configuraciones de píxeles están presentes, una pequeña rotación alrededor del eje óptico causaría la distribución de probabilidad de las configuraciones de píxel medidos, para cambiar drásticamente. Esto puede resultar en una respuesta de esquina incorrecta y lenta.

El aprendizaje de árbol de decisión que utiliza el algoritmo FAST construye un árbol ternario con posibles estados de píxeles "más oscuro", "brillante" y "similar". En cada etapa de aprendizaje, las dos preguntas, "¿es más brillante?" y "¿es más oscuro?", se aplican en todos los píxeles restantes y el máximo se elige. El estado de cada píxel puede ser una de cuatro posibilidades: desconocido (u), más oscuro (d), más brillante (b) o similar (s). Esto es lo que llamamos una combinación de N estados para una configuración de píxeles. El tamaño del espacio de configuración es, por lo tanto 4 N, que produce $4^{16} \approx 4 \cdot 10^{9}$ configuraciones posibles para N = 16.

Espacio de configuración de un árbol binario de búsqueda para AGAST

En lugar de considerar solo un espacio de configuración restringido, como en FAST, este enfoque tiene un espacio de configuración más detallada, con el fin de proporcionar una solución más eficiente. Tiene en cuenta para evaluar una sola pregunta cada vez. La idea es elegir uno de los píxeles para evaluar y preguntar por la posición. La pregunta evalúa el píxel dado, y la respuesta se utiliza para decidir el siguiente píxel y la pregunta a consultar. Este enfoque busca una esquina que reduce al atravesar un árbol de decisión binario. Se requiere para especificar qué píxel consultar y el tipo de pregunta a utilizar. En consecuencia, el espacio de configuración se incrementa en la adición de otros dos estados: "No más brillante" (b) y "No más oscuro" (d). Usando la notación para el estado de un píxel en relación con el núcleo n, denotada por $n \rightarrow x$, se asigna como sigue:

$$S_{n \rightarrow n} = \begin{cases} d, & I_{n \rightarrow n} < I_n - t & (darker) \\ \bar{d}, & I_{n \rightarrow x} \not< I_n - t \wedge S'_{n \rightarrow x} = u & (not\ darker) \\ s, & I_{n \rightarrow x} \not< I_n - t \wedge S'_{n \rightarrow x} = \bar{b} & (similar) \\ s, & I_{n \rightarrow x} \not> I_n - t \wedge S'_{n \rightarrow x} = \bar{d} & (similar) \\ \bar{b}, & I_{n \rightarrow x} \not< I_n + t \wedge S'_{n \rightarrow x} = u & (not\ brighter) \\ b, & I_{n \rightarrow x} \not< I_n + t & (brighter) \end{cases} \quad \text{(AII.23)}$$

donde $S_{n \rightarrow x}$ es el estado anterior, que es el brillo de un píxel y significa que el estado aún se desconoce. Esto da como resultado una representación de árbol binario, en oposición a un árbol ternario. Lo que permite una sola evaluación en cada nodo. Tenga en cuenta que esto aumenta el tamaño del espacio de configuración para 6 N, que rinde $6^{16} \approx 2 \cdot 10^{12}$ nodos posibles para N=16

Asociado con cada rama de nuestro árbol existe un costo de procesamiento, lo que representa el coste computacional en el equipo que realiza las operaciones. Estos costos varían debido a los diferentes tiempos de acceso de memoria. Especificamos estos de la siguiente manera:

- c_R: coste de acceso al registro (segunda comparación del último píxel probado).
- c_C: coste de acceso a la caché (test de un píxel en la misma fila).
- c_H: coste de acceso a memoria (test de ningún otro píxel).

Además para cada píxel se requieren costes de procesamiento adicional debido a las operaciones de evaluación.

Los algoritmos de inteligencia artificial como ID3, realizan bastante mal los árboles de decisión óptima. Sin embargo, el problema de encontrar un árbol de este tipo es un problema bien estudiado, donde se ha demostrado que el óptimo global es un problema de tipo NP-completo.

Para obtener el árbol de decisión óptimo se implementa un algoritmo que es similar al método de inducción hacia atrás. Se explora todo el espacio de configuración a partir de la raíz del árbol de decisión. Se sabe que ninguno de los píxeles es conocido. Los nodos del árbol se forman mediante la evaluación recursiva con una posible pregunta en un píxel dado. Se explora el espacio de configuración (usando la primera búsqueda en profundidad) hasta que una hoja se encuentra, en donde se define como el primer nodo de camino que cumple o no puede cumplir los criterios de esquina AST. El coste de una hoja dada es cero, mientras que el coste de un nodo interno es c_P, es determinado, recogiendo el costo mínimo calculado para cada par de hijos c^+ y c^-, en representación de los resultados positivos y negativos de un test, por

$$c_P = \min_{\{(c_+, c_-)\}} C_{C+} + p_{C+}\, c_T + c_{C-}\, c_T = c_{C+} + c_{c-} + p_P c_t \quad \text{(AII.24)}$$

donde c_T representa el costo de la evaluación píxeles con $c_T \in \{c_R, c_C, c_M\}$ y el p_P, p_{C+} y p_{C-} son las probabilidades de las configuraciones de píxeles en los nodos padre e hijo respectivamente .

Usando esta técnica de programación dinámica nos permite encontrar el árbol de decisión para una óptima AST (OAST) de manera eficiente. El árbol de decisión resultante puede ser optimizado para diferentes c_R, c_C y c_M, sino las probabilidades arbitrarias para cada configuración de píxel.

Cada imagen, independiente a la escena, está formada por zonas homogéneas y (o) desordenadas que representan superficies uniformes o regiones estructuradas con textura. En lugar de aprender la distribución de las configuraciones de píxeles de imágenes de entrenamiento, como en FAST, una primera generalización sería aprender la probabilidad de regiones estructuradas, homogéneas y optimizar el árbol de decisión de acuerdo con esta distribución. El árbol resultante es

afinado y optimizado para la escena preparada, mientras que es invariante a rotaciones de la cámara. La probabilidad de una imagen para ser uniforme, se realiza modelado por la probabilidad de un estado pixel a ser similar al núcleo (p_s). Los estados "más brillantes", "más oscuros" reflejan los estados, por ejemplo: un píxel más brillante en el patrón de prueba evaluará el píxel núcleo actual como más oscuro en cuanto se convierte en el píxel central.

Debido a esta duplicación de los estados "más brillante" y "oscuro" se supone que tienen la misma probabilidad (p_{bd}), que se elige para resumir a uno con p_s mediante la expresión($p_s + 2p_{bd} = 1$). Por lo tanto, la probabilidad de una configuración de píxel p_X se puede calcular como sigue:

$$p_x = \prod_{i=1}^{N} con\ p_i = \begin{pmatrix} 1 & para\ S_{n \to i} = u \\ p_s & para\ S_{n \to i} = s \\ p_{bd} & para\ S_{n \to x} = d \vee S_{n \to x} = b \\ p_{bd} + p_s & para\ S_{n \to x} = \bar{d} \vee S_{n \to x} = \bar{b} \end{pmatrix} \quad \text{(AII.25)}$$

La distribución de probabilidad de la configuración píxel tiene una distribución de trinomio con las probabilidades p_s y dos veces p_{bd}. Tenga en cuenta que los estados $\bar{d}, \bar{b},$ no son muestras de esta distribución, sino que representan un conjunto de dos y tres muestras, respectivamente. Si bien este enfoque ofrece una buena solución para los medios de ensayo, no es genérica como FAST, que tiene que aprender para cada escena específica donde se aplica.

Para conseguir una solución más eficiente y genérica, el algoritmo se adapta automáticamente a la zona que actualmente está procesado, es decir, se cambia entre los árboles de decisión que están optimizados para el área específica. La idea es construir, por ejemplo: dos árboles y especializar uno para homogéneo y otro para las regiones estructurados sobre la base de un valor pequeño y grande para p_s.

Al final de cada ruta de decisión, cuando se cumple el criterio de esquina o no puede ser cumplida completamente, se salta al árbol especializado apropiado se realiza en base a la configuración de píxel de esta hoja (como se muestra en la figura AII.6) . Este intercambio entre los árboles especializados de decisiones viene con ningún coste adicional, la evaluación del nodo hoja se realiza fuera de línea cuando se genera el árbol especializado. De esta manera, el AST se adapta a cada sección de la imagen de forma dinámica y se aumenta su rendimiento, para una escena arbitraria. Cualquier aprendizaje se vuelve innecesario.

Debido a un cambio entre los árboles sin coste, solo se puede realizar en una hoja, la adaptación tiene un retraso de una prueba. El único caso de adaptación y el test de segmento acelerada genérica (AGAST) sería menos eficiente que FAST, es si el ambiente cambiaría de ser homogénea a la inversa estructurado y viceversa en píxeles consecutivos. Esto no es posible en la

práctica, debido al efecto de reflejo de píxeles diferentes como se describió anteriormente. Sin embargo, las imágenes naturales no tienen una distribución de brillo al azar, sino que son más bien divididos en regiones desordenadas y uniformes. Si los árboles de decisión pueden ser fuertemente equilibradas variando p_s, también se pueden usar más de dos árboles ponderados diferentes.

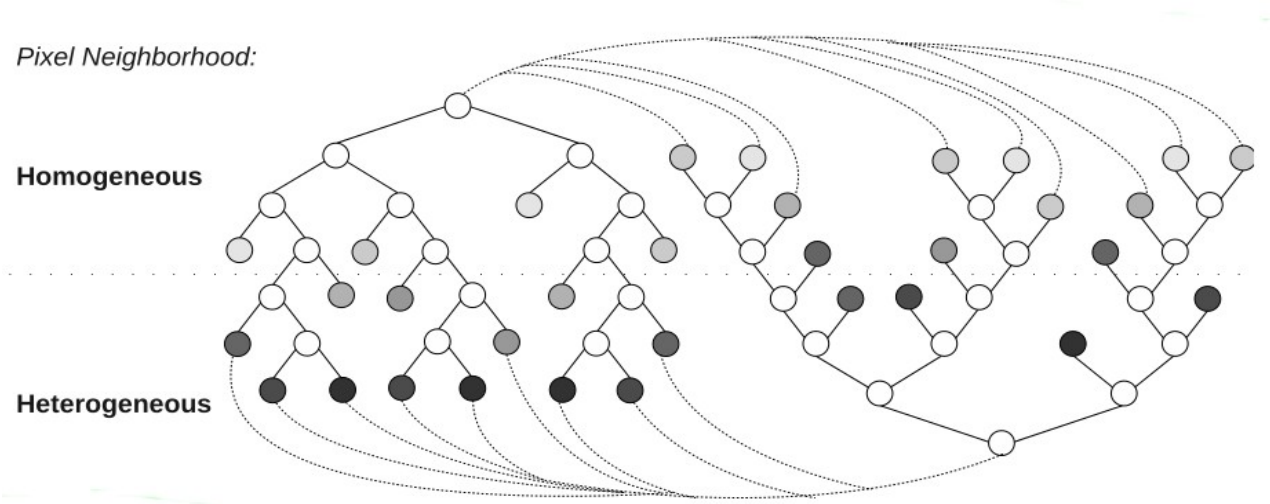


Figura AII.6: Árbol de decisión de AGAST

Anexo III

Este anexo se realiza un estudio de las potencialidades en diferentes librerías, las que se puede hacer uso para el registro de nubes de puntos. Estos están centrados principalmente sobre la librería Point Cloud Library, que está orientada para el manejo de nubes de puntos, no es impedimento para que se pueda utilizar otras librerías que no estén orientadas para este tipo de formato de datos.

En la librería PCL se plantea la solución de registrar nubes de puntos, el esquema de la figura AIII.1 presenta una vaga descripción del procedimiento, con los módulos que deben utilizarse para obtener un registro completo. La librería PCL dispone de muchas clases y dependencias, para trabajar con nubes de puntos, pero solo se estudiaran aquellas que necesitamos usar para perseguir nuestro objetivo. En la figura AIII.1 no se concretan las clases que se deben utilizar, si conocemos en que módulos tenemos las clases y funciones necesarias.

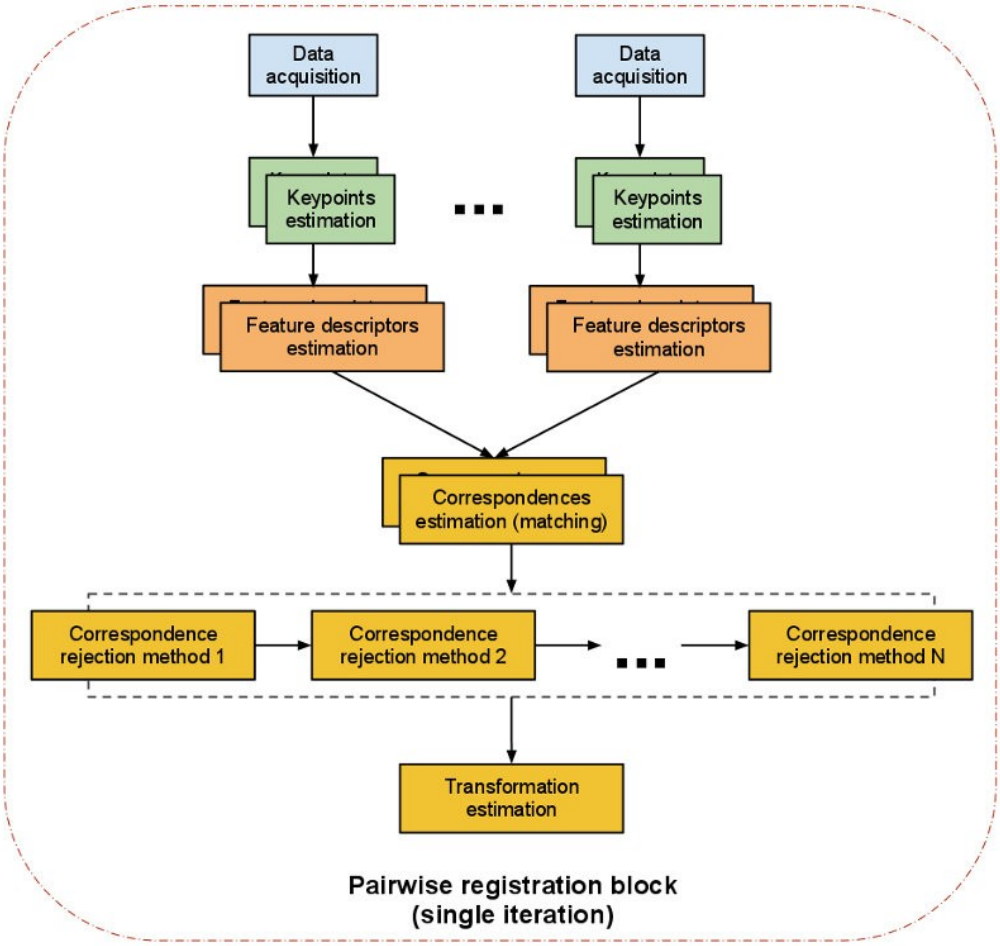


Figura AIII.1: Esquema de registro de nubes según desarrolladores de PCL(cortesía de http://pointclouds.org)

1 Características de la nube

Las nubes aportan una información de la escena obtenida a través de un sensor. Los datos recogidos están sujetos a diferentes errores, que se trasladan a las características, estos se calculan con la vecindad de puntos. Se estudia cuáles son los problemas más evidentes y cómo afectan a la información de la nube.

1.1 Nubes de puntos

Las nubes de puntos son la imagen que entrega nuestro sensor al sistema, donde se produce todo el ruido en el proceso, teniendo en cuenta la calidad de los sensores con los que vamos a trabajar. Los sensores que entregan los datos con mayor precisión, son lo que captaron las imágenes mediante tecnología láser, como es el sensor LIDAR. Los sensores con tecnología láser son muy caros, hay que sacrificar la calidad de la imagen de entrada, con un sensor que sea más asequible, que si bien retorna una imagen con mayor ruido, nos permite trabajar con menores costes. En todo caso, estas tareas también serán necesarias para sensores con mayor o menor calidad.

El sensor kinect, con el cual se trabajará en la mayor parte del proyecto, es un sensor de codificación de luz o visión estereoscópica, los más asequibles en el mercado, tiene los siguientes problemas para la utilización:

1. Errores relativos diferentes en función de la distancia de la superficie.

2. Los bordes producen la obtención de puntos fallidos.

3. En caso de los sensores de codificación de luz son sensibles a otras fuentes de luz.

Para el caso de imágenes de Kinect® debemos de tener en cuenta que debido a la limitada precisión obtenida en la tecnología de luz codificada, podemos apreciar en la figura AIII.3 el efecto, donde las nubes de puntos forman multitud de pequeños planos, cuando estos no están perpendicular al sensor, como se explica en la figura AIII.2 donde se representa la realidad y lo capturado por el sensor Kinect.

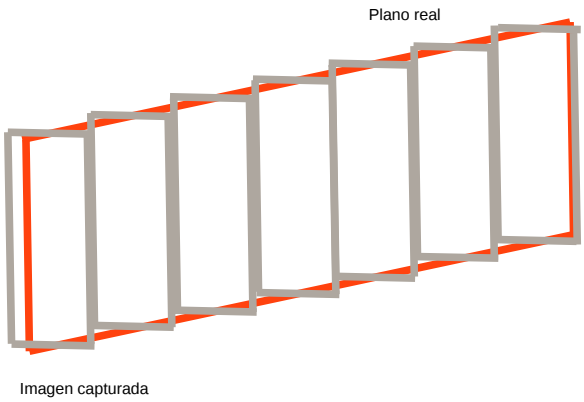


Figura AIII.2: Visualización esquemática del efecto de multiplanos en captura de nubes oblicuas.

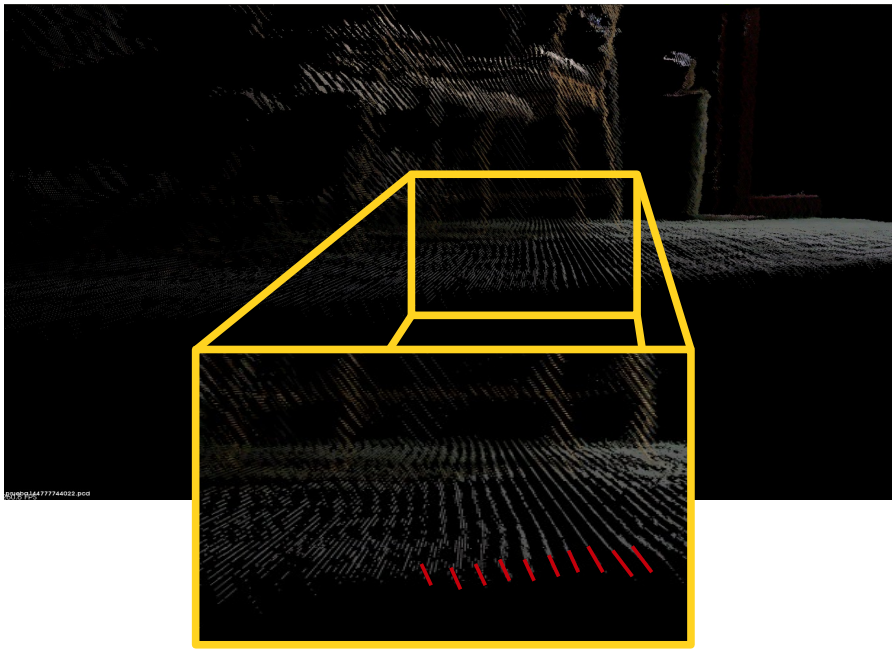


Figura AIII.3: Visualización del efecto de multiplanos en una nube real capturada con Kinect.

Por otro lado los sensores tienen las siguientes ventajas:

1. Los tiempos de muestreo son suficientemente pequeños, si deseamos hacer un mapeo a una velocidad moderada.

2. Los datos se entregan de forma ordenada, bajo un criterio matricial.

Cuando las escenas son capturadas con sensores con tecnología Time-of-Flight o LIDAR, donde se captura todo el entorno con una visión de 360º, la imagen tiene gran precisión en la captura de los datos. El funcionamiento de este sistema se realiza con muestreos de ángulos, capturando la distancia a la superficie de cada incremento de ángulo. En la figura AIII.4 se muestra la captura de un sensor de este tipo.

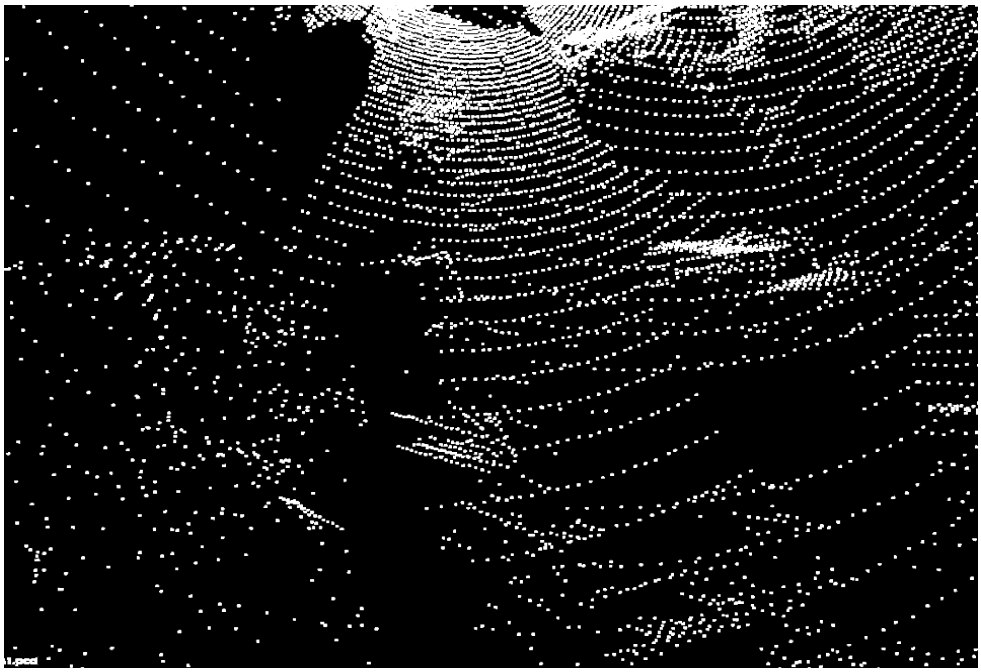


Figura AIII.4: Visualización de círculos concéntricos en una nube capturada mediante sensor LIDAR

Como se observa en la figura AIII.4, se crean en la nube cadenas de puntos concéntricas al punto de captura. Estas coronas tienen una distancias entre los puntos vecinos que la forman, es inferior a los vecinos más cercanos de las coronas concéntricas. Además las superficies más cercanas tienen mayor densidad de puntos. Por lo que existe el problema de no tener una distribución y resolución uniforme, lo cual es un problema para aplicar las máscaras.

Ambos tipos de sensores son susceptibles de ser mejorados, hace que los procesos de obtención de información, se desvirtúen por los efectos que se producen al ponderar los puntos, sin tener en cuenta los efectos. Para evitarlo se tomarán máscaras lo más grande posible, lo que hará perder en muchos casos el localismo de las características.

1.2 Vector normal

En la librería PCL existen dos clases para el cálculo del vector normal, se distinguen por realizar en una de las clases mediante multihilos. El resultado final en ambos caso es el mismo, sin embargo si se dispone de capacidad de multiprocesamiento los tiempos de cómputo se reducen en proporción a los hilos disponibles. Las clases son las siguientes:

1. **pcl::NormalEstimation**: estimación simple.

2. **pcl::NormalEstimationOMP**: estimación multihilo.

El vector normal es una característica local de la nube. Se define mediante los puntos vecinos. El parámetro más importante es la máscara, que tomará los vecinos más cercanos, y para que se pueda ajustar a la nube de puntos se tiene en cuenta la resolución media del conjunto. A partir de esta métrica, se puede imponer un valor relativo. Es importante que la máscaras sea lo suficientemente grande para atenuar el efecto del ruido y los defectos del sensor.

En la figura AIII.5 vemos como tomar un radio que abarque un conjunto muy pequeño, intentando tomar en mayor medida el efecto local, se generan de forma poca uniforme para los planos.

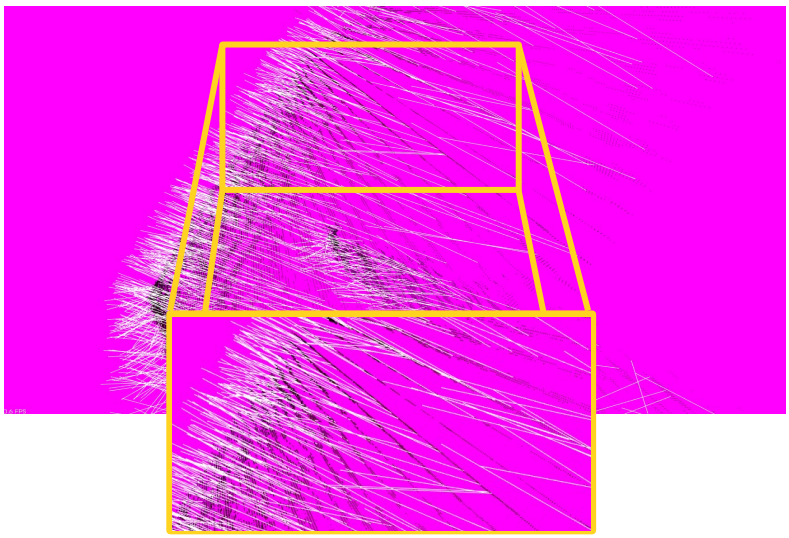


Figura AIII.5: Visualización de normales con radio umbral pequeño.

En el caso de las nubes obtenidas mediante Kinect, si la máscara es muy pequeña, el efecto de escalonamiento se trasladaría a las normales, generando vectores como los observados en la figura AIII.6. no perpendicular a la superficie real. El efecto es tan local que no representan la realidad capturada, por lo que no representa la información que se requiere.

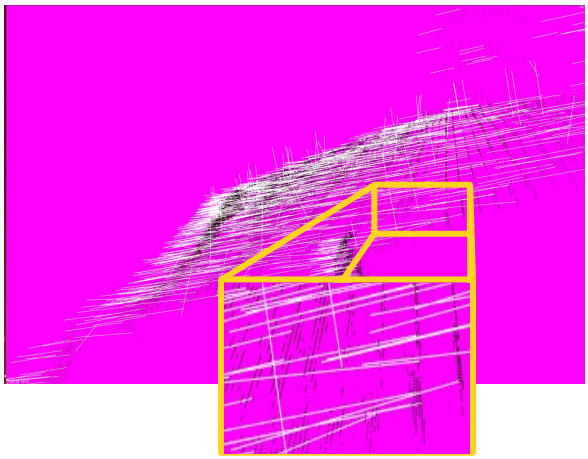


Figura AIII.6: Visualización de normales con radio umbral lo suficientemente pequeño para que las normales tengan en cuenta un solo de los multiplanos.

Tomar una máscara excesivamente grande repercutirá en el localismo del parámetro que se calcula, perdiendo información. Tal efecto es difícil de ponderar, si bien lo cambios producidos entre la normal de los distintos vecinos de una zona singular no aportarían la información del cambio. Por el contrario, en las zonas uniformes se obtiene un efecto deseado de uniformidad. Para que el efecto local de escalonamiento de los puntos, no sea demasiado fuerte y se obtengan vectores consistente a nivel macroscópico se aplicarán máscaras lo suficiente grande, deberá tener un aspecto con vectores lo más paralelo posible, cuando se toman puntos pertenecientes a un mismo plano, como se muestra en la siguiente imagen, donde la máscara es de 500 puntos vecinos, aunque siempre es posible tener vectores que son afectados por el ruido de la imagen, pero no se pueden aplicar máscaras excesivamente grandes, pues no se calcularía una normal que represente un efecto local como se pretende.

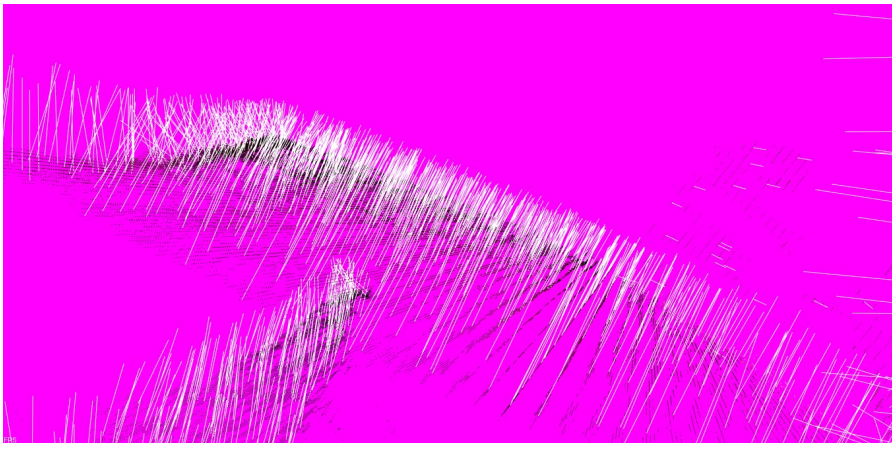


Figura AIII.7: Visualización de normales con radio umbral grande con 500 puntos de máscara.

Para realizar los cálculos, se aplicará una medida relativa, sobre un valor que tenga en cuenta las características de los puntos de la nube. Como valor se tomará la resolución media, es un parámetro fácil de calcular y servirá para otros procesos.

El valor de la resolución media es un valor relativamente pequeño, mide la distancia media entre el puntos más cercanos. Es decir, si aplicamos una máscara , en la resolución media habrá puntos que tengan varios vecinos para el cálculo de su normal, pero otros no dispondrán de vecinos con el que tomar la medida.

El valor de resolución media está calculado de forma estadística, es necesario recurrir a modelos de probabilidad para establecer el umbral, este nos garantiza que bajo él existen los valores suficientes, para calcular el valor local,con una muestra representativa. Son necesarios al menos tres puntos en la muestra, pero el objetivo es que la muestra más pequeña tenga al menos 5 puntos.

La distribución normal o de Laplace-Gauss N(μ,σ) servirá para estudiar que la probabilidad del umbral dado tenga al menos un punto cercano o más. Teniendo en cuenta que μ es el valor del punto de consulta y σ se le asignará el valor de la resolución media. Si estas medidas las tipificamos podemos estudiar la distribución de Laplace N(0,1), de dónde queremos saber: cuál es la distancia a la media, que abarca una probabilidad significativa para que el umbral tenga al menos un punto. Para ellos utilizando las tablas de la curva Normal de la siguiente figura.

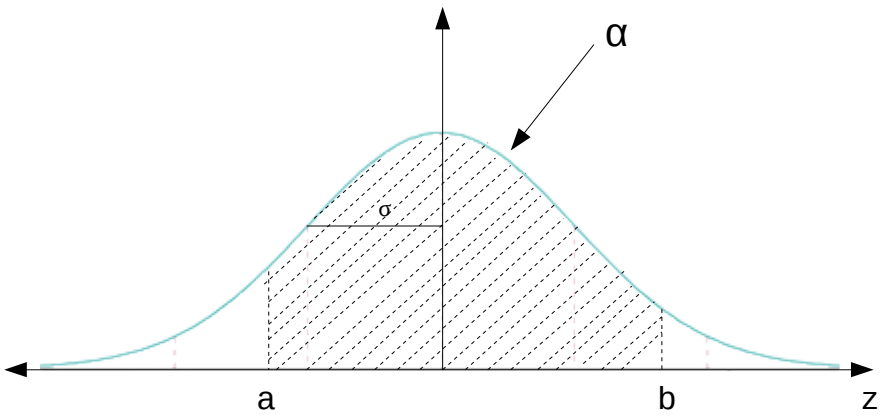


Figura AIII.8: Tipo de área que tenemos en cuenta para estimar el umbral de las normales.

De la cual se puede extraer el área con la siguiente expresión:

$$\alpha = \int_{a}^{b} \frac{1}{\sqrt{2\pi}} e^{-z^2/2} dz \qquad \text{(AIII.1)}$$

Como estamos trabajando con valores tipificados, el valor obtenido será el relativo al umbral de la resolución media, por lo que no necesitamos reconvertir. Además como los valores son siempre positivos tomaremos un área simétrica, como simplificación de nuestro caso, donde a=-b, Por lo que se tomará un valor igual a la distancia entre a y b. Para obtener un alto nivel de significación se tomará el mayor área de la curva, ocupa un intervalo de (-∞,∞), nos ajustaremos a un valor de α=0.99, entrando en tablas corresponde a un valor aproximado de z=3,3. Por tanto,el

valor aproximado necesario para obtener al menos dos puntos cercanos es de 6,6 que será redondeado a 7.

Con un umbral de 7 veces la resolución media tenemos la posibilidad de encontrar en una nube de 302700 puntos, una media de 144 puntos, donde no habrá vecinos con los que calcular las normales. La cantidad de puntos cautivos de vecinos supone un 0,46% de los puntos, una cantidad que se puede asumir.

Por último, hay que tener en cuenta que asociado al vector normal, la librería PCL calcula el valor de la curvatura máxima. Este parámetro, representa el mayor cambio que se produce en el punto, puede ser de gran ayuda para la estrategia de diferentes algoritmos, se implementan, pues ayuda a conocer el entorno de de la vecindad, sin necesidad de comparar entre sí los vectores de los puntos vecinos.

2 Filtros

Los filtros implementados en PCL se han creado para disminuir la resolución o para atenuar el ruido o errores en la captura de la imagen, sin disminuir la resolución. En nuestro proyecto se realizará el filtrado de las nubes con el propósito de disminuir los tiempos de cómputo, pues las máscaras aplicadas abarcaran un menor número de puntos.

En el estudio de los filtros, se observará cómo afecta la aplicación de los filtros sobre nubes de puntos obtenidas con sensor Kinect. Los valores se aplicarán sobre la nube modelo, de la figura AIII.8 Los efectos son conocidos, sin embargo no hay estudios en los que se calibran los que realiza sobre la nube de salida. En cuanto los tiempos de cómputo para el procesamiento, debido al bajo nivel de operaciones necesarias para aceptar, rechazar o calcular la nueva posición, se considera despreciable.

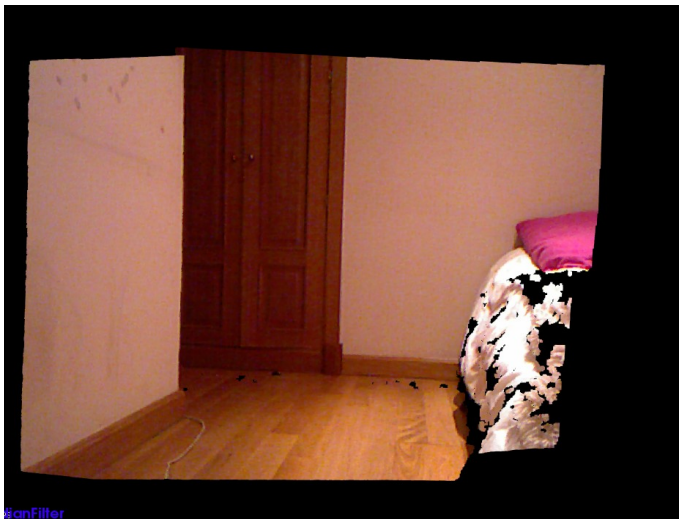


Figura AIII.8: Imagen de ejemplo para aplicar filtros

pcl::Filter

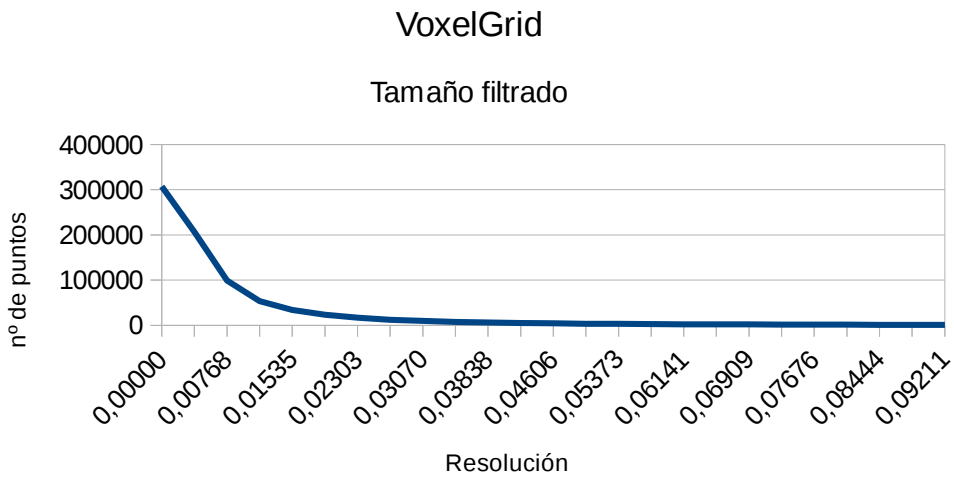
Esta es la clase base con la que trabajan el resto de clases del módulo de filtros, en la introducción de datos al algoritmo que procesa la nube de puntos. Esta clase no realiza ninguna modificación sobre la nube.

pcl::removeNaNFromPointCloud

Esta clase realiza la eliminación de datos sin valor, denominados NaN, se reduce los tiempos de cómputo por la toma de todos los datos, para los distintos cálculos o comparaciones. Sin embargo, no hay pérdida de información debida a las superficies. En los casos de los punto con datos que contienen solo valores de RGB queda incompleta al necesitar el valor de profundidad. Cuando se realiza el filtro sobre una nube tomada con kinect, la reducción de puntos queda bastante sujeta a la cantidad de bordes que hay en la escena, sin embargo se promedia una reducción de alrededor del 20-50% de los datos. Si se desea estudiar características 2D de la nube, como si se tratase de una imagen digital, no se puede aplicar este filtro porque se pierde el orden matricial obtenido en la captura.

pcl::VoxelGrid

Se aplica el filtro VoxelGrid para reducir el tamaño de la nube de puntos, con un volumen menor de datos. El filtro segmenta el espacio en cajas donde ubica los puntos para estudiar el centroide de los puntos pertenecientes a cada caja. Si se aplican tamaños de cajas muy pequeños solamente se borrarán los datos que no tienen valores de profundidad. En la gráfica siguientese observa cómo se reduce la nube de puntos en relación al tamaño del cubo aplicado.



Es apreciable en la gráfica como la nube decrece rápidamente con valores cercanos a la resolución media. Para valores 5 veces la resolución media el volumen de puntos decrece en menor medida. Es recomendable, no aplicar valores muy grandes. En algunos casos podría perderse mucha información. Cuanto más grande sea el volumen de la red que se aplica, más representativo será el punto del conjunto al que sustituye. El punto final representa el centroide de los puntos a los que pertenece, como se muestra en la siguiente figura donde se resalta los puntos seleccionado por el filtro, se observa que si bien disminuye la resolución no se desvirtúa la nube con deformaciones.

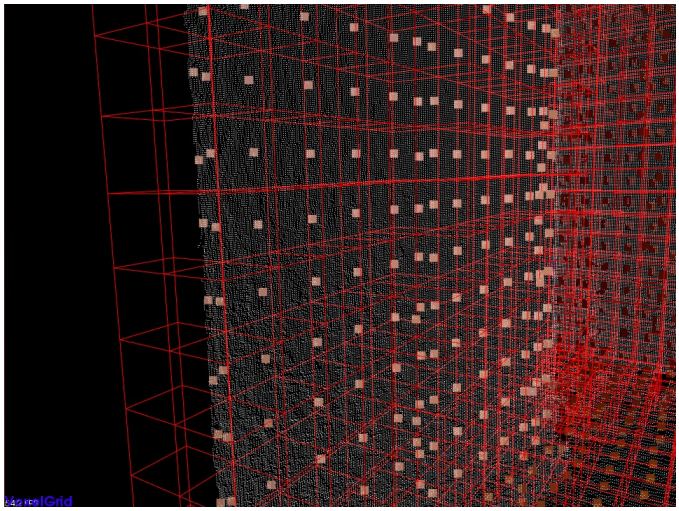
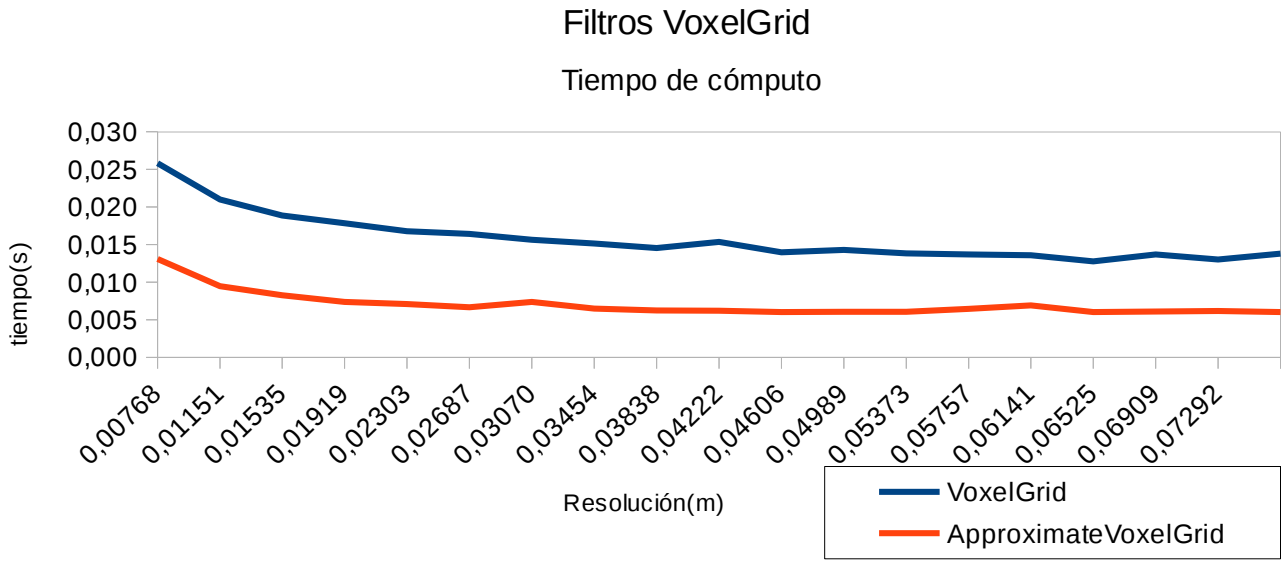


Figura AIII.9: Visualización de los "voxel" para filtros Voxel Grid, con los puntos de salida resaltados.

pcl::ApproximateVoxelGrid

Esta clase implementa un algoritmos de reducción de puntos similar a VoxelGrid, se diferencia por ser mucho más rápido para generar la nube de salida. Sabemos que, para conseguir una mayor velocidad no utiliza procesos multihilos, como ocurre en otras clases. Sin embargo, como se puede

observar en la siguiente gráfica, donde se comparan los tiempos de esta clase con VoxelGrid, se observamos que los tiempos se reducen a la mitad. No varía considerablemente con la resolución exigida.



Entendemos que la nube obtenida establece los centroides en los cubos que se predeterminan. Sin embargo, este algoritmo no cumple las exigencias al igual que VoxelGrid. Si bien la mayor parte de los cubos generan un centroide, otros cubos pueden tener varios puntos de salida. Este fenómeno se puede visualizar en la figura AIII.10.

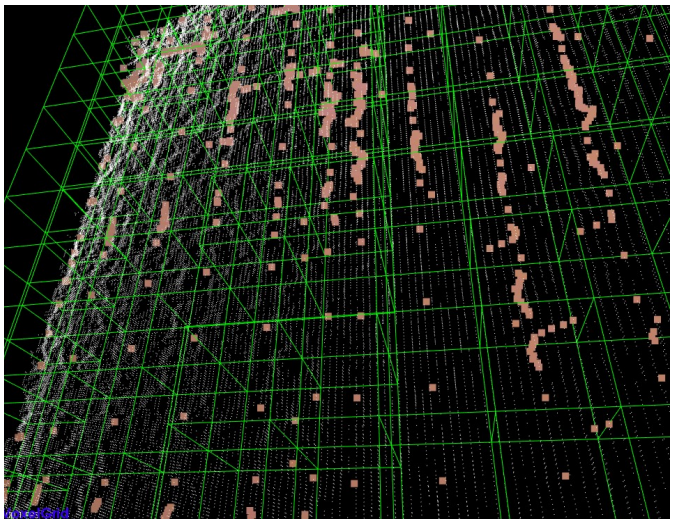
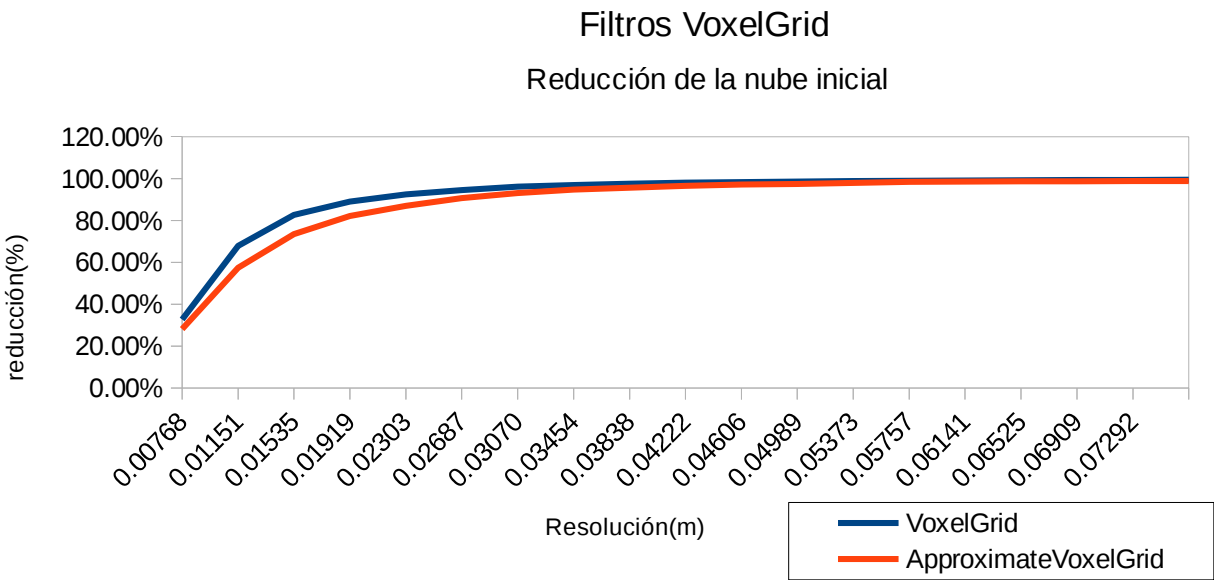


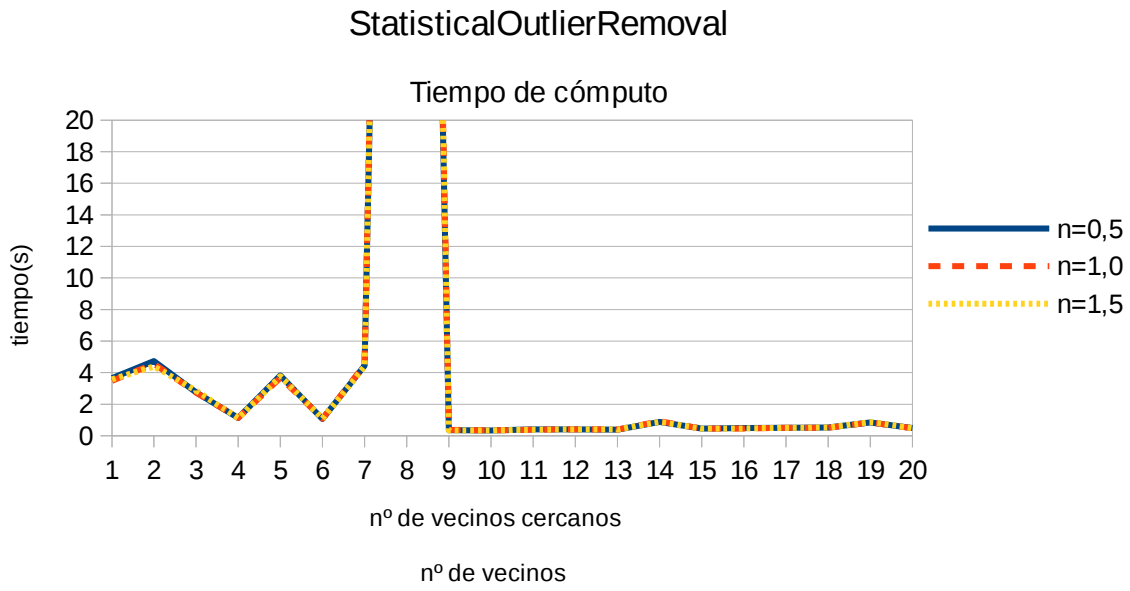
Figura AIII.10: Visualización de los "voxel" para filtros “Approximate Voxel Grid”, con los puntos de salida resaltados.

Si los cubos tienen un número mayor de puntos que en VoxelGrid, donde se generaba un centroide por cubo, ahora la clase estudiada genera nubes algo mayores, con una distribución no uniforme. La reducción de puntos entre ambas clases no es significativamente mayor, si bien tampoco se puede considerar tiempos tan pequeños que la reducción a la mitad sea significativa.



pcl::StatisticalOutlierRemoval

Este método aplica métodos estadísticos para la discriminación de puntos de la nube. Itera dos veces sobre la nube. En una primera iteración se calcula la distancia media, que tienen cada punto con los “k” vecinos más cercanos. El valor “k” se puede ajustar con la función setMeanK(k). Se determina la desviación típica y estándar(stddev) de todas las medidas, con el fin de establecer un umbral, este se halla de la siguiente manera: media + stddev_mult * stddev. El valor stddev_mult representa un valor estándar. Si se aplica como valor 1, genera aproximadamente un 84% de inlier. Este parámetro si se puede modificar mediante la función setStddevMulThresh(). Aquellos valores que superen el valor del umbral calculado por la clase imponiendo los vecinos que se toman y el multiplicador de la desviación, son rechazados. Se realizará un sondeo para nuestra imagen modelo. El sistema será más restrictivo con los puntos en función del múltiplo de desviación que impongamos, como podemos ver en la siguiente gráfica, con apenas variar el tamaño de la nube, con la elección del número de puntos cercanos. Si observamos que se estabiliza para valores de k=5, por lo que es evidente que las medias son más estables para valores de k>4.



Sin embargo, es importante estudiar cómo afecta en el tiempo de cómputo una mayor toma de puntos vecinos, como se puede observar en la siguiente gráfica. El sistema tienes problemas para estudiar los 8 vecinos más cercanos, y aunque podría tener problemas para estudiar mayor volumen de vecinos, lo cierto es que su comportamiento es mejor, por lo que se evitarán valores inferiores a k<9.

pcl::MedianFilter

La aplicación de este filtro no supone la reducción de la nube de puntos, por lo que transforma los datos atípicos, realizando la mediana con la máscara de vecinos más cercanos. Es bastante eficiente porque no necesita iterar varias veces sobre la imagen.

Para realizar la mediana se toma el dato de profundidad z, sin modificar (x , y). Debe atenuarse los efectos de ruido por impulso, aquel que sitúa al punto mucho más lejos de lo que debería capturar. Sin embargo, como se puede apreciar en la siguiente figura no se evita el efecto de escalonamiento, están los puntos estratificados por capas.

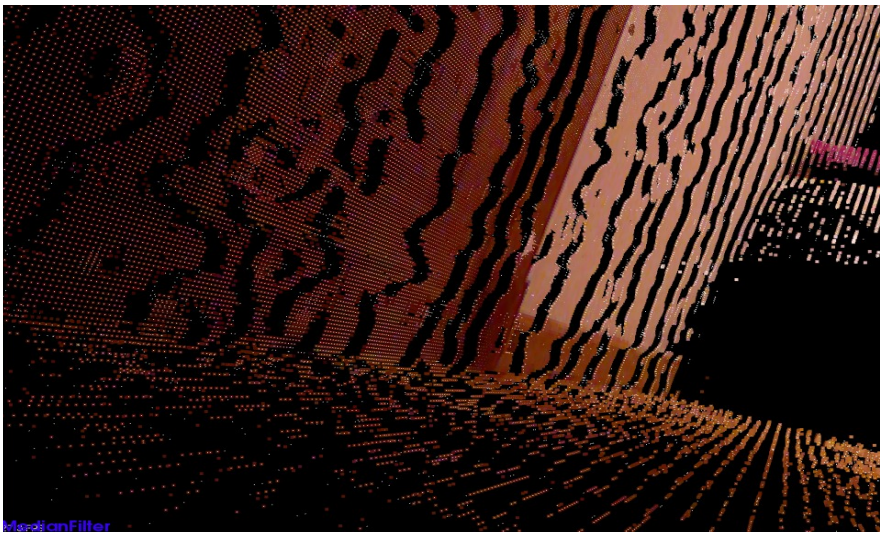


Figura AIII.11: Nube de puntos filtrada mediante “Median Filter”

Los tiempos de cómputo son bastante pequeños, alrededor de 0.01seg. Tiene la ventaja de no desordenar la matriz, por la que se ordena la matriz de nube de puntos, aunque por otro lado no admite nubes que estén desordenadas.

3 Módulo de Keypoints

Los keypoints son puntos que tienen un alto valor, porque ofrece información de cambio con respecto a los puntos de su entorno. Su principal característica es que son únicos, por lo que su presencia relativa es baja. En esta sección estudiaremos los algoritmos que han sido implementados en PCL, el capítulo anterior se explicó teóricamente su funcionamiento.

Para el estudio de keypoint repasamos sus característica y definimos las métricas que serán estudiadas. Teniendo en cuenta que debe cumplir las siguientes condiciones:

1. Escasez: la nube que debe entregar los algoritmos debe contener un conjunto relativamente muy pequeño de número de puntos. Se estudiará el volumen de puntos entregado, siendo mejores los que entreguen pocos puntos.

2. Distinción: este parámetro nos indica el nivel de información, pues si consideramos que estamos filtrando la nube según su información una métrica que tiene en cuenta esto será el volumen de datos, que será mayor cuanto menor información se necesite para ser aceptado como conjunto de keypoints.

3. Repetibilidad: Este parámetro nos indica como es de estable el mecanismo de obtención, si dos nubes de puntos iguales deben entregar los mismo keypoints, este parámetro se estudiará la sensibilidad al ruido y su sensibilidad frente a pequeños cambios.


Figura AIII.12: magen modelo para estudio de keypoints.

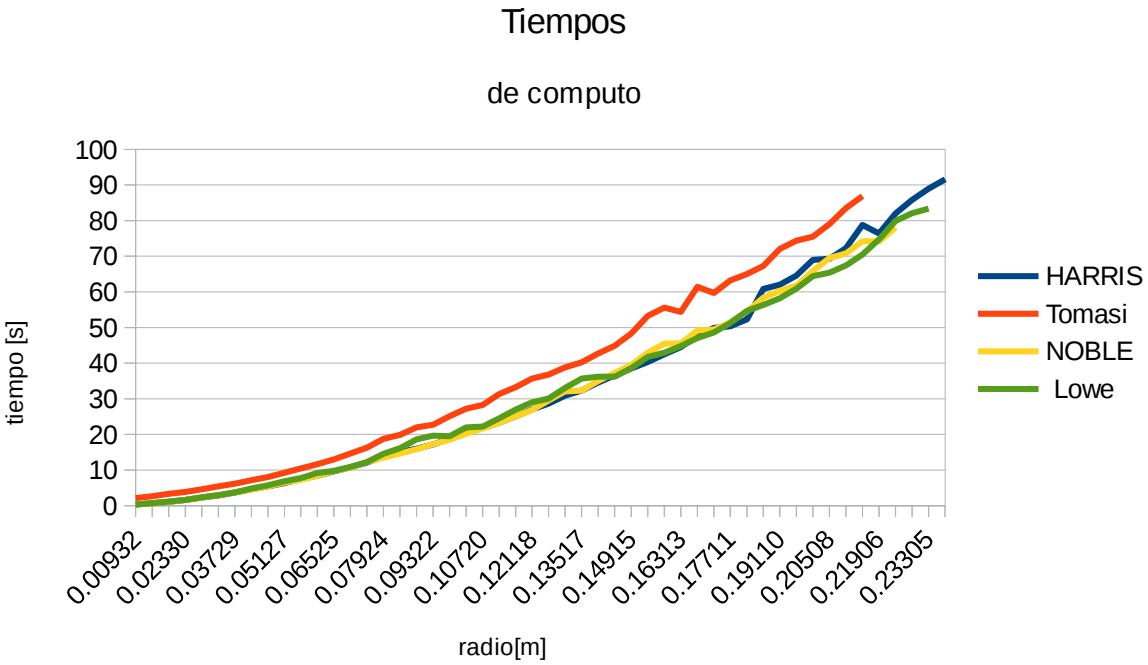
En el procedimiento de estudio, se estudiarán los 3 parámetros más importantes que representan una mejor idoneidad de los keypoint. Para esto se escoge una imagen común en un entornos interior, con alguna figura geométrica compuesta que provoque singularidades con entornos suaves como paredes. En nuestro estudio tomaremos la imagen de un comedor como se muestra en la figura AIII.12 la cual no se le aplicará ningún tipo de tratamiento previo que la modifique como un filtro. La escena modelo ha sido capturada con varias nubes de puntos, para obtener una misma escena con diferente ruido. Es necesario tener en cuenta la sensibilidad que tienen los algoritmos frente a las variaciones de ruido, donde todos los parámetros que se pueden controlar, como la iluminación y posición del sensor. No han sido modificados.

También es necesario estudiar la sensibilidad que se produce, frente a pequeños cambios de umbral. Se estudiará la repetibilidad entre los mismos puntos de interés a diferentes umbrales, realizando correspondencias restrictivas, obteniendo valores absolutos y los relativos frente al volumen total de keypoints. La relación entre las correspondencias entre los diferentes keypoints es un buen estimador de la repetibilidad. Pueden existir errores en alguna de las correspondencia. Para realizar el estudio de keypoint se tomará como paso entre umbrales. En los casos de detectores escala, la resolución media de la nube.

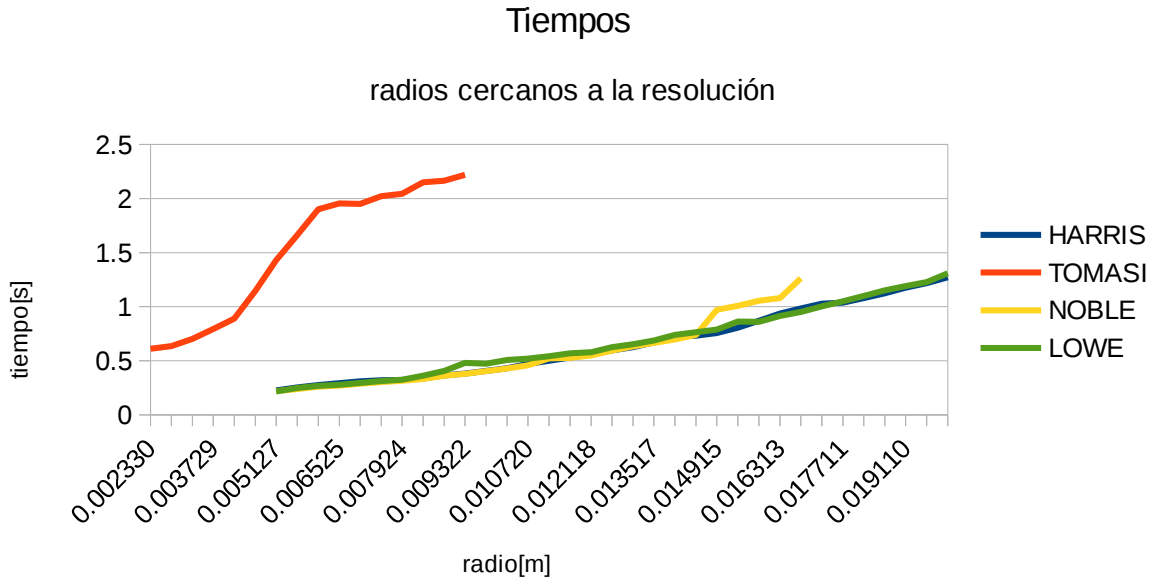
3.1 Estudio del método Harris 3D.

Este método de cálculo de puntos de interés, tiene varios enfoques para obtenerlo. Estudiando el método, analizaremos de forma general, la respuesta de los diferentes enfoques. El estudio necesita analizar el tiempo de respuesta para el cálculo de los keypoint. Se realiza en tiempo real.

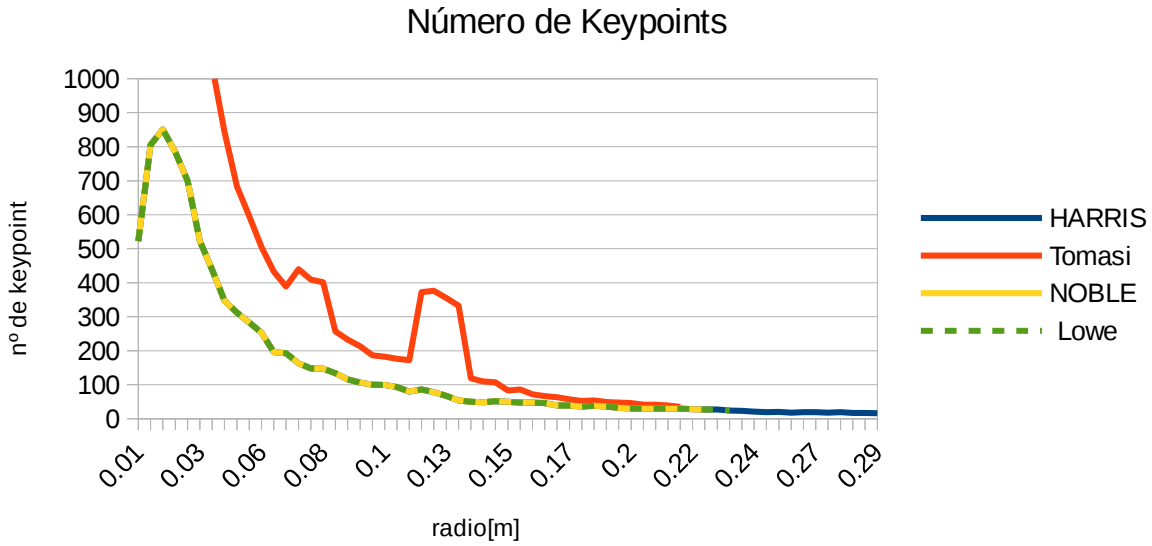
Sabemos, la entrada que más influye en el tiempo de cálculo es la cantidad de puntos en la nube. El radio agrupara los puntos que serán escogidos para la matriz de covarianza. En el siguiente gráfico se muestra la respuesta del tiempo para diferentes radios.



Se observa que los tiempos de cómputo para todos los métodos son muy parecidos para los diferentes radios, debido a que los cálculos que realiza cada método son bastante similares. Sin embargo, la gráfica tiene un amplio rango de radios, en relación con la resolución. Si son muy grandes pueden llegar a superar los 60 segundos de procesamiento. En la siguiente gráfica, focalizamos los radios para los cuales los tiempos de cómputo son inferior a un segundo obteniendo la siguiente gráfica:



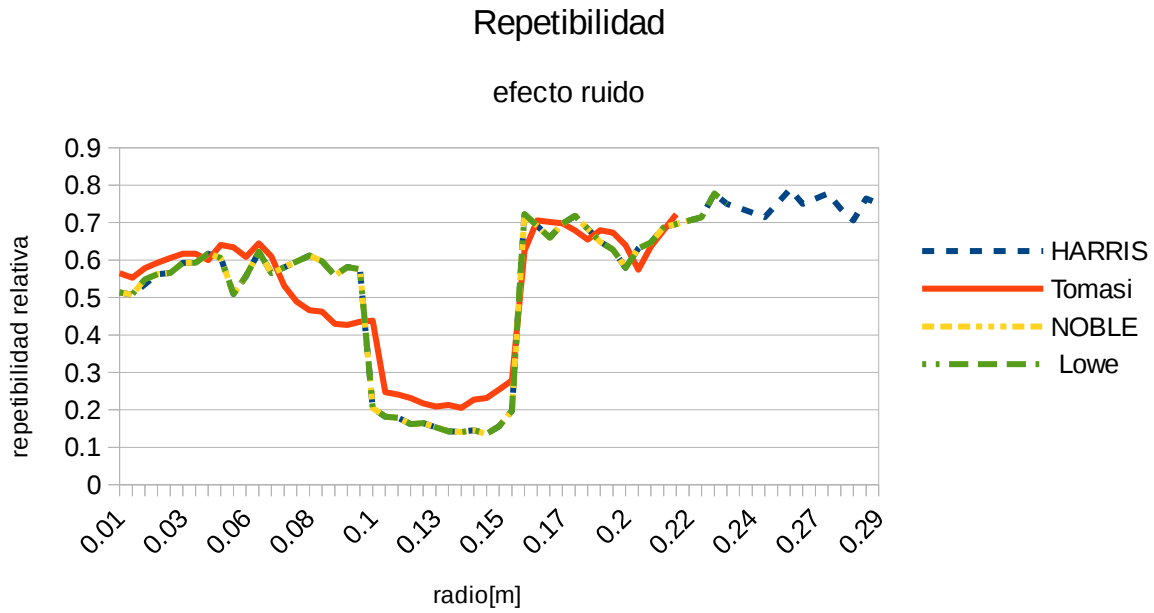
Los tiempos de cómputo siguen siendo muy similares para todos los métodos, excepto para TOMASI, que supera dos segundos para radios cercanos al de resolución(resolución ≈ 0,005 m). Para pequeñas resoluciones tenemos tiempos de cómputo no muy elevados, en relación a los datos que se procesan.



Conociendo la capacidad que cada método tiene para buscar los puntos de interés, estos filtran la imagen para encontrar puntos que deben tener una información local relevante. Siendo suficientes para realizar otras operaciones. En la gráfica anterior se cuantifica el volumen de keypoints que entrega cada método para diferentes radios, teniendo en cuenta que solo se aplica un umbral de rechazo pequeño para que los keypoint no estén demasiado juntos

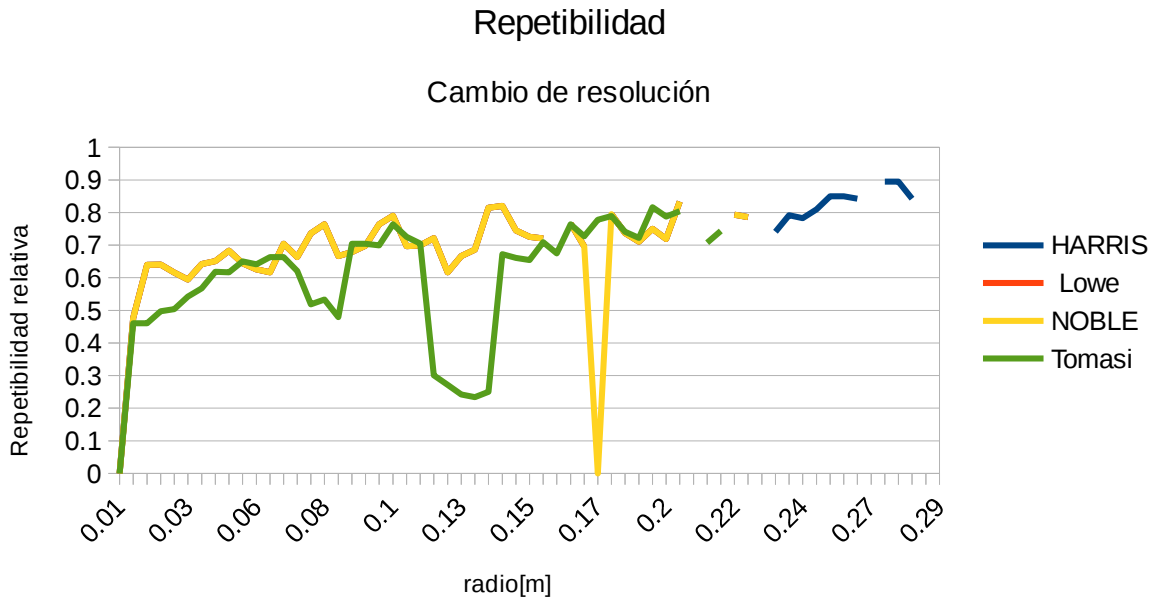
En la respuesta que se obtienen para los diferentes umbrales, es conveniente aplicar valores altos para obtener nubes con un volumen menor. Aunque esta decisión es contrapuesta a la obtenida en las respuesta de tiempo frente a radio umbral.

Otra característica que debe ser tenida en cuenta es la repetibilidad de los keypoint, pues no indicará lo bondadoso que es el método para encontrar los puntos que realmente son de interés, donde suponemos que existe una variación de ruido.



Como podemos comprobar la repetibilidad relativa para bajas resoluciones no es muy buena, apenas se repiten el 60% de los puntos, se deduce que estos métodos son bastantes sensibles al ruido.

Analizaremos el efecto de la repetibilidad por variaciones del radio de búsqueda, para esto tomaremos como paso de variación, la resolución de la imagen y comparemos los keypoints obtenidos con dos radios de búsqueda diferentes que tendrán un incremento entre ambos igual a la resolución de la imagen.

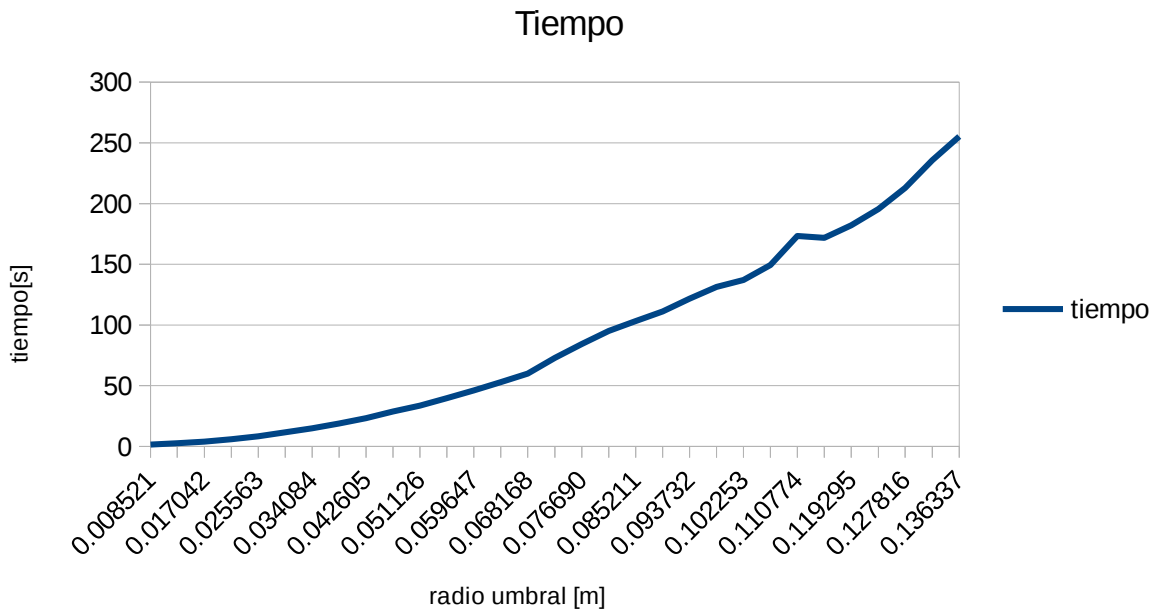


Observamos que para una misma imagen,el cambio de radio de búsqueda afecta notablemente en la repetibilidad de los keypoints.

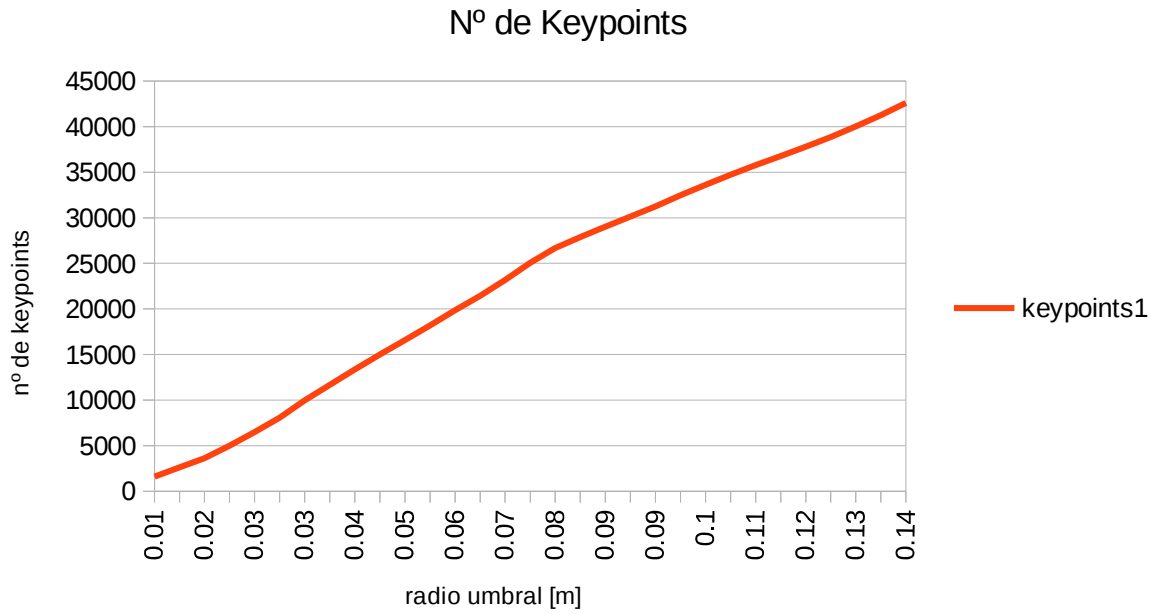
3.2 Estudio del método SUSAN

Para el estudio de del método SUSAN hay tener en cuenta los parámetros sobre la imagen de dicho método. Primero se estudiará los cambios que se producen en el brillo y el ángulo que se forman en estos contornos. Todos estos parámetros son estudiados de forma local, siendo el núcleo, el píxel o punto estudiado. El resto de los puntos que son circundantes, serán buscados según un umbral que se impone. El parámetro más sensible en cambios de tiempo y número de keypoint encontrados será el umbral.

Primero veremos cómo influye el radio de umbral sobre el tiempo. Se toma valores recomendados en la biblioteca que utilizamos(son cambios de brillo de 10 unidades sobre los 255, y como ángulo umbral 0.001) obtenido los siguientes resultados:

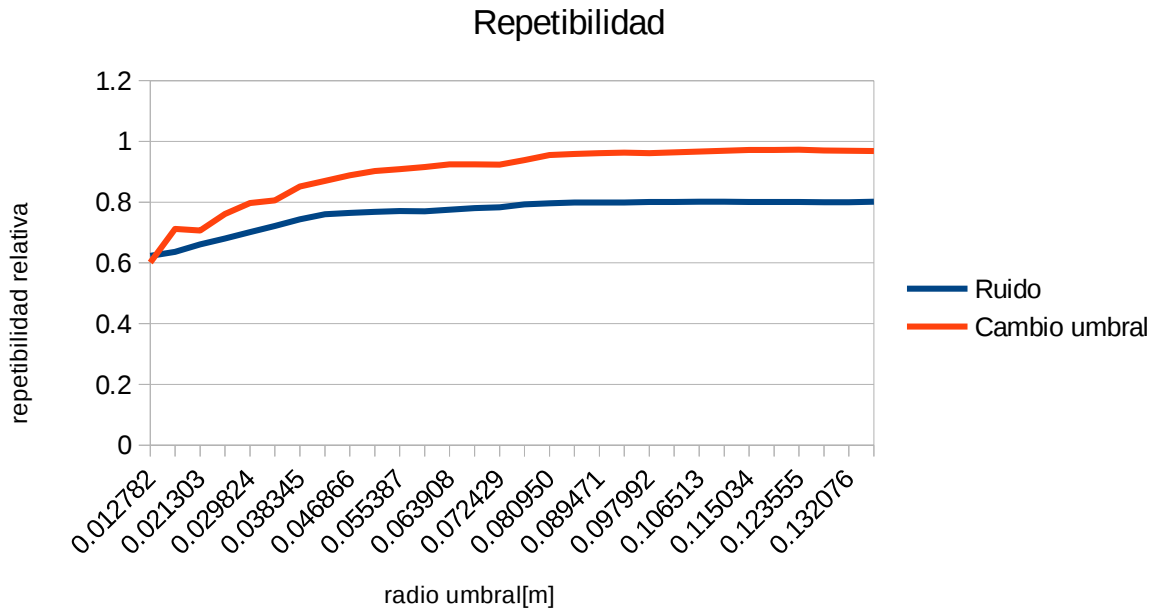


Como vemos en la gráfica, los tiempos son directamente proporcionales al radio de umbral, superando tiempos de un segundo de cómputo. Por lo tanto es un método bastante lento. Otro factor importante es la cantidad de keypoint entregados por el método. Desearemos detectar todas las esquinas que se producen por el cambio de brillo en la imagen.



Como podemos ver en la gráfica, hay una relación directa entre el radio umbral y el número de keypoints obtenidos. Se deduce que los parámetros locales son más evidentes aplicando umbrales mayores. Para umbrales pequeños el número de puntos es aceptable, pero con umbrales algo mayores la cantidad de puntos entregados son excesivos, comúnmente no suele suceder así por lo que es destacable que aparezcan pocos puntos de interés para umbrales pequeños. Los costes de computación son elevados para llegar al volumen de keypoints.

Por último, para todo método de cálculo de keypoints que se precie, la repetibilidad de los puntos calculados, siguiendo procedimientos de estudios anteriores, calcularemos la repetibilidad relativa para dos imágenes con diferentes ruido y para una misma imagen con cambio de radio de umbral, este cambio será proporcional a la resolución.



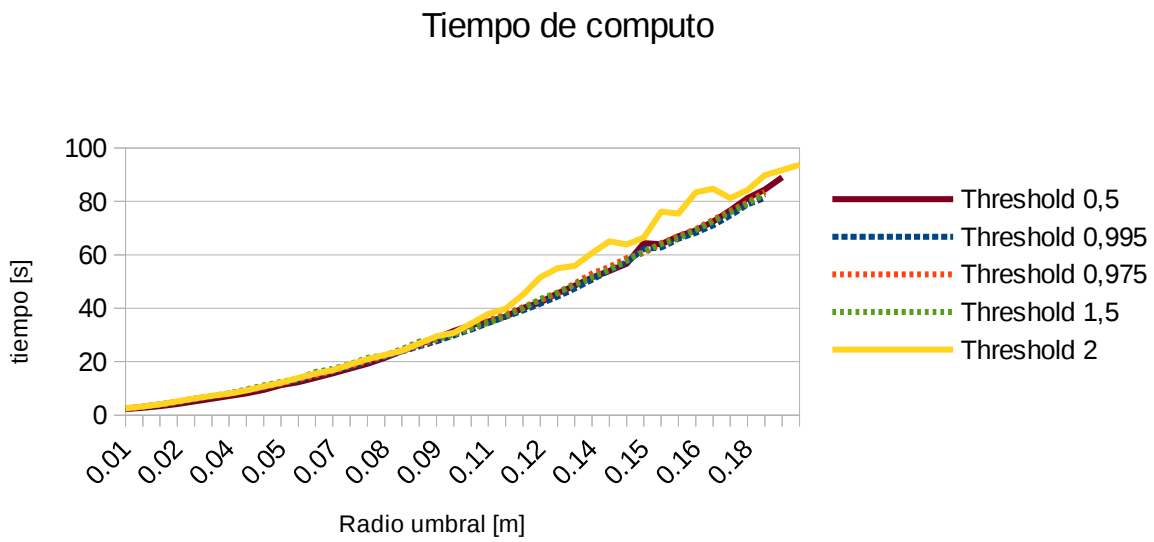
Como podemos observar los cambios debido al ruido influyen negativamente en relación con los de cambio de umbral. Se concluye que tiene una repetibilidad aceptable para pequeños umbrales. Con estos, el número de puntos no se reducen significativamente, y dejar los posibles problemas a resolver incompletos.

Este método es bastante ineficiente para la obtención de puntos de interés, pues solo es capaz de cumplir con el parámetro de repetibilidad. Los de tiempo de cómputo y de volumen de puntos son tan elevados que anula los buenos resultados en la repetibilidad.

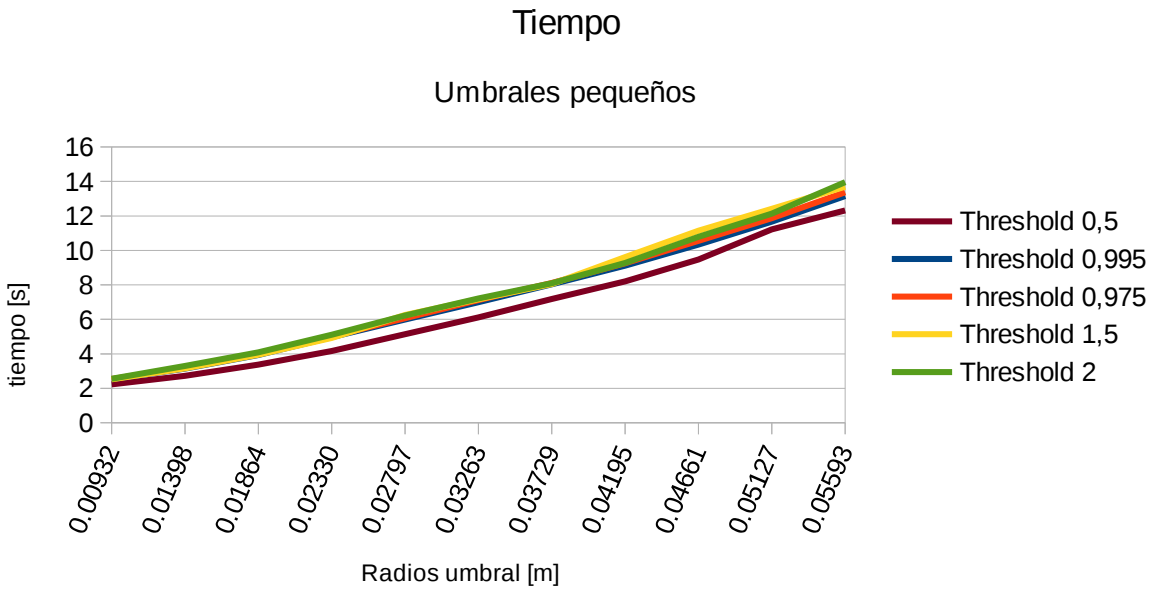
3.3 firma intrínseca de forma

Los parámetros que podemos controlar como la diferencias entre autovectores, el umbral de búsqueda y supresión de puntos muy cercanos. Analizaremos principalmente el umbral de búsqueda como parámetro que produce mayor sensibilidad en los datos de salida, los tiempos de cómputo se analizarán diferentes diferencias de autovalores, con el fin de encontrar diferencias en la repetibilidad de los keypoints.

En la siguiente gráfica, se analizan los tiempo de cómputo para varias diferencias de autovalores distintas, con un amplio rango de umbrales. Se observa que apenas hay cambios entre las diferencias de autovalores, el proceso más costoso es calcular la matriz de covarianza, al concurrir más puntos puntos para radios mayores. El proceso para encontrar los keypoints es más lento. Las posibles divergencias de tiempos de cómputo se debe a las condiciones del procesador, no al aumento del autovalor estudiado.

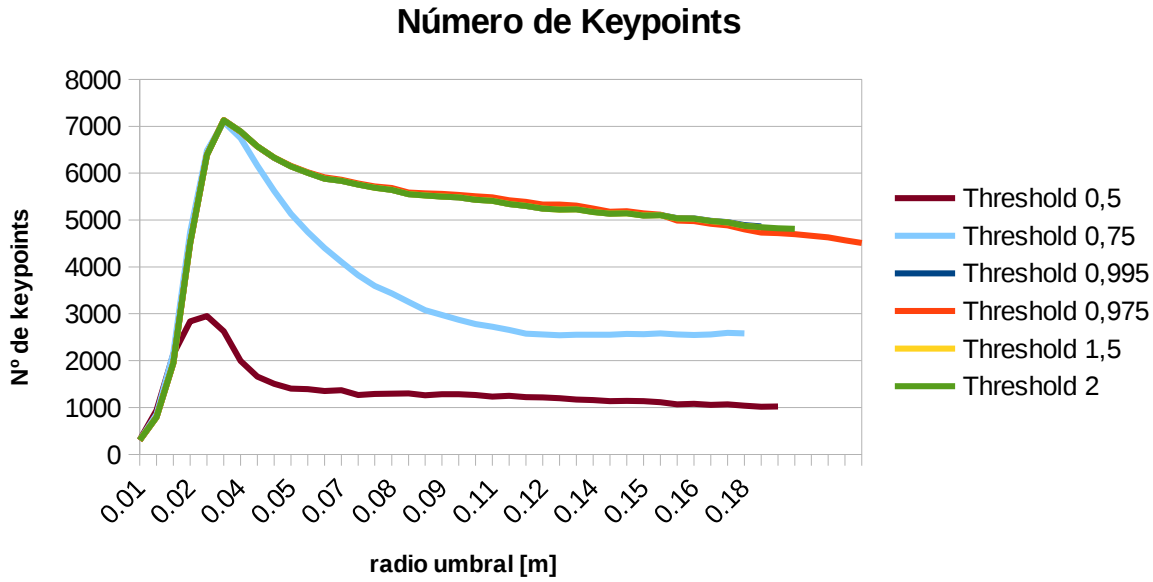


Teniendo en cuenta umbrales pequeños y cercanos a la resolución media de la imagen y haciendo un pequeño zoom en la gráfica del tiempo de cómputo en el origen de la mismo. Para diferentes autovalores no se determina sensibilidad que nos permita hacer una elección



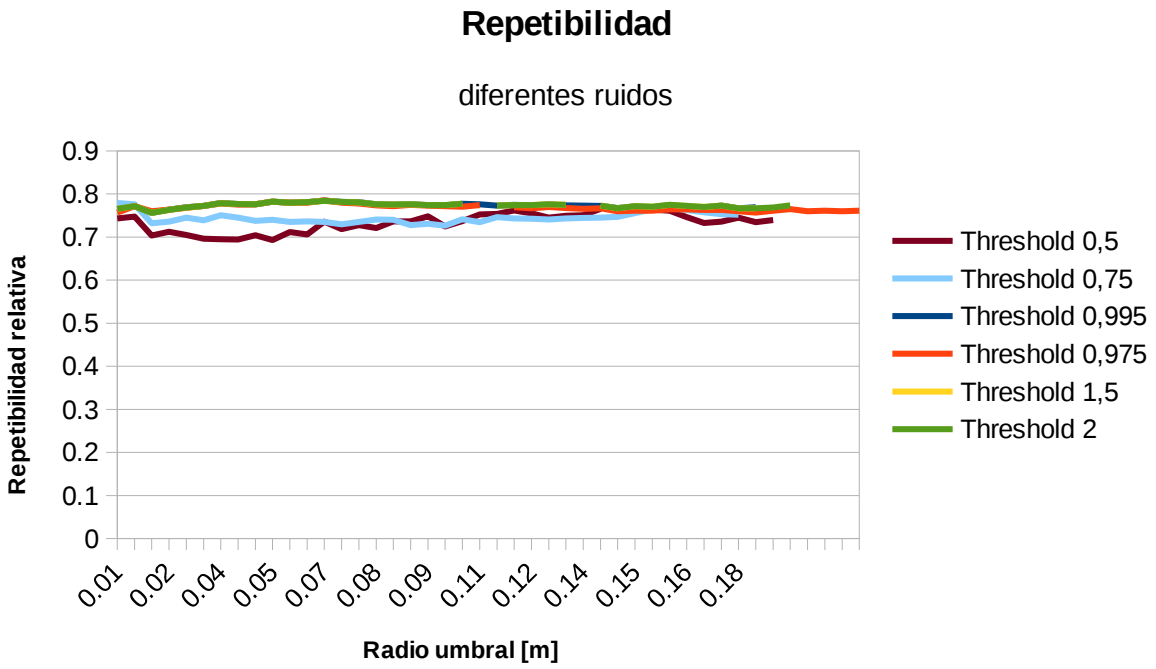
Hay que estudiar el número de puntos de interés que se obtienen, lo que indicará como es de restrictivo el método. Se observa en el número de keypoints entregados. Hay abundancia de puntos de interés de salida, suponemos que para las diferencias de autovalores introducido que cubre un

espectro,no muy amplio, pero sí representativo, se presupone que la firma intrínseca no tiene porque cumplir condiciones de esquinas u otras singularidades. Según las diferencias de autovalores encontraremos una forma, que se repetirá en función del número de keypoints de salida.

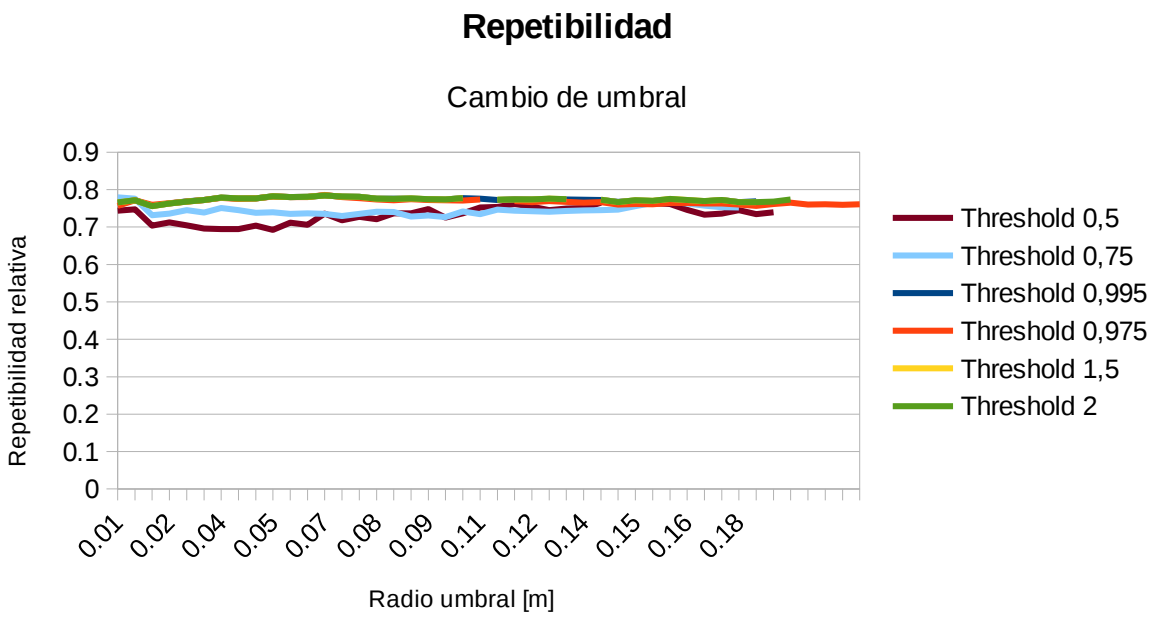


Destacar que la restricción superior se produce para diferencias inferiores a la unidad. Para mayores, las condiciones se relajan y no producen una sensibilidad sobre la cantidad de puntos de interés de salida.

En la gráfica de repetibilidad relativa para diferentes ruidos, tiene unos parámetros aceptables que supera valores relativos de 0.75 para diferencias de autovalores superiores a la unidad. Unido a la abundante cantidad de puntos, facilita las tareas de resolución de los problemas.



Estudiamos la repetibilidad por cambios en el radio de umbralización. En esta gráfica se observa que para radios de umbral pequeños la repetibilidad es aceptable, para todo el rango estudiado, si bien se considera que podría ser mejorable.



Los datos obtenidos demuestran que la calidad del método es aceptable, y los tiempos de procesamiento no crecen fuertemente con un cambio de umbral, siguen siendo elevados para la tarea que nos proponemos realizar.

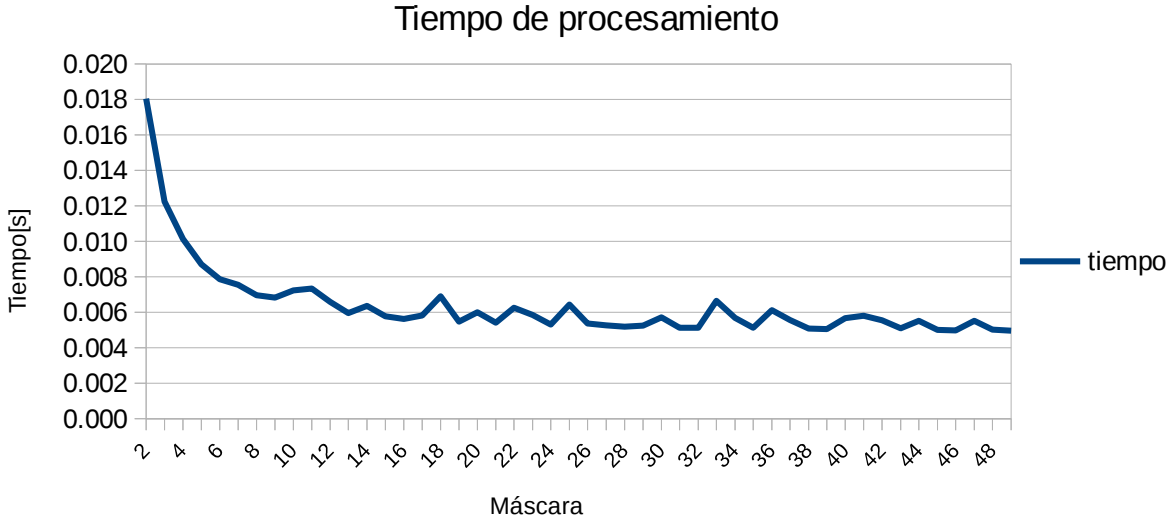
3.4 AGAST

Los métodos con adaptación de escala, necesitan convertir las nubes de puntos en una imagen fotográfica, de forma está contenida en una matriz ordenada en filas y columnas, con cada píxel en una de las posiciones de la matriz. Se consigue trabajar en dos dimensiones. Realizando tareas que se utilizan convencionalmente en procesamiento de imágenes digitales.

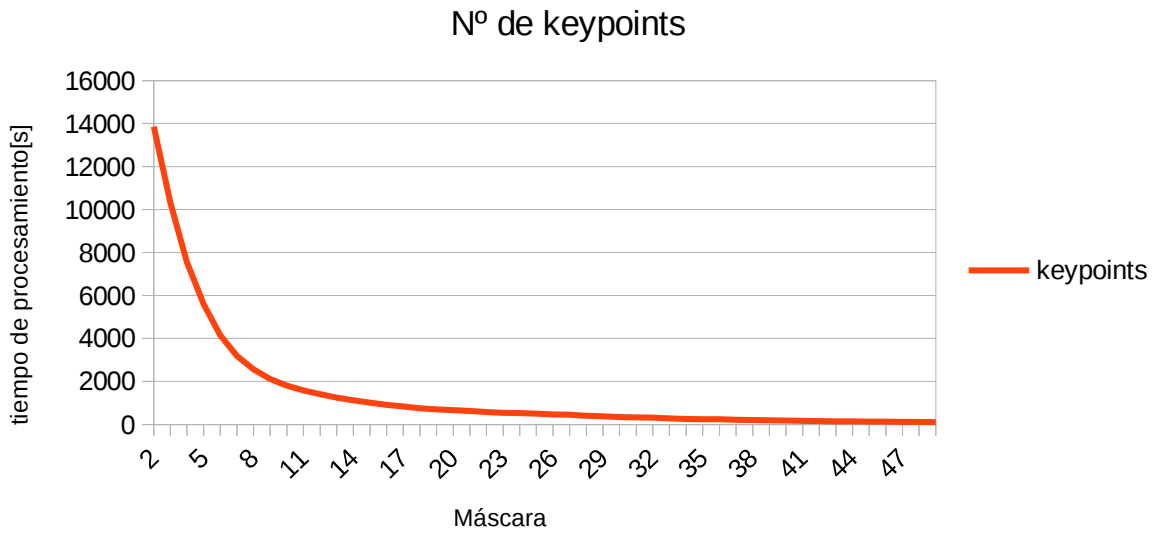
Trabajando sobre la base de 2 dimensiones, no se usan umbrales sino máscaras matriciales, donde se tiene en cuenta las posiciones más cercanas, que en algunos casos no serán puntos cercanos en la nube. La máscara se establece con números enteros, que indican cuántos píxeles cercanos se incluyen en el proceso.

Los datos de entrada, como se ha comentado anteriormente, vendrán dados en parámetros de 3D y los procesaremos para convertirlos en imágenes digitales en 2D. Se podría inducir a una pérdida información, pero se mantiene la nube original, para buscar los índices que pertenecen de cada píxel con cada punto, que se considerará keypoint, para emparejarlos con procedimientos y algoritmos establecidos para nubes de puntos.

En la siguiente gráfica, observamos que lo tiempos de cómputo son considerablemente bajos. Los mecanismo de 2D son bastante sencillos y requieren pocas operaciones, para decidir si un punto es considerado keypoint o no.

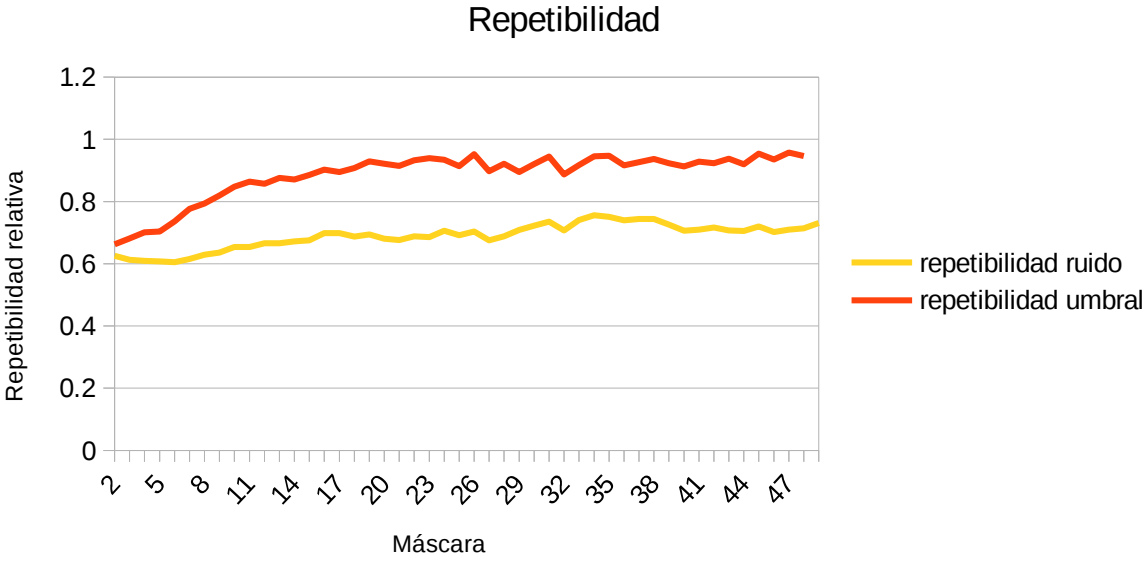


Este método emplea tiempos de cómputo muy bajos, que curiosamente bajan con el empleo de máscaras más grandes, lo que debería de suponer un número de operaciones mayor, y se obtienen tiempos similares a partir máscaras =20 en adelante.



La máscara es más restrictiva cuanto más grande es y los puntos de interés generados son cada vez son menores. En el gráfico, no llegan a anularse pero si fuera más restrictivo, en algún momento la salida de puntos de interés se anularía, porque se le exigiría una cantidad excesiva de información que ningún punto es capaz de aportar.

En la siguiente gráfica se muestran los datos de repetibilidad para diferentes máscaras.



Los valores de repetibilidad que se obtienen son aceptable, al igual que en los métodos anteriores, si bien es más sensible al ruido. Los cambios de umbral responde mejor cuanto más restrictivo son los parámetros de análisis. El método es bastante eficiente por tener bajos tiempos de cómputo, con respuestas similares a las que se generan con los métodos de escala fija.

3.5 Comparación de métodos Keypoints

Los métodos de obtención que han sido descritos anteriormente, donde se explicó cuáles son los parámetros de estudio para una mejor fiabilidad de estos, además de los tiempos de cómputo de los diferentes umbrales. Todos los métodos, a pesar de utilizar diferentes enfoques, se obtienen datos de repetibilidad similares. Sin embargo los tiempos varían considerablemente.

En la siguiente tabla mostramos un resumen con los mejores valores de cada método para la repetibilidad y tiempos de cómputo para el umbral al que corresponde.

Agast_2d

umbral threshold	tiempo	Keypoints 1	kepoints2	repetibilidad repetibilidad ru	varianza	repetibilidad umbral
34	0,0056855	258	253	0,755814	0,000883454	0,917266
48	0,0050205	112	109	0,714286	0,00289344	0,957265
49	0,0049665	108	103	0,731481	0,0039806	0,946429

método Harris-Tomasi

umbral radius	threshold	tiempo	keypoints1	kepoints2	repetibilidad repetibilidad_	varianza	rep_nube_2
0,00932179	0,0186436	2,16321	9327	9311	0,565134	1,65E+011	0
0,191097	0,0186436	72,0192	47	49	0,673469	0,00529975	0,816327
0,214401	0,0186436	86,8412	35	36	0,722222	0,00644777	0,74359

método Harris-harris

umbral radius	threshold	tiempo	keypoints1	kepoints2	repetibilidad repetibilidad_	varianza	repetibilidad_nube_2
0,00932179	0,0186436	0,387458	522	495	0,51341	33842,1	0
0,256349	0,0186436	106,329	18	19	0,789474	0,00311168	0,85
0,270332	0,0186436	122,05	18	17	0,777778	0,012501	0,894737

método Haris-Lowe

umbral radius	threshold	tiempo	keypoints1	kepoints2	repetibilidad repetibilidad_	varianza	repetibilidad_nube_2
0,00932179	0,0186436	0,389011	522	495	0,51341	33842,1	0
0,200418	0,0186436	64,3782	29	38	0,631579	0,0032896	0,833333
0,228384	0,0186436	82,0582	27	23	0,777778	0,0128782	0,7632275

método Harris-Noble

umbral radius	threshold	tiempo	keypoints1	kepoints2	repetibilidad repetibilidad_	varianza	repetibilidad_nube_2
0,00932179	0,0186436	0,384578	522	495	0,51341	33842,1	0
0,15847	0,0186436	45,4599	47	46	0,723404	0,00326862	0,7429785
0,200418	0,0186436	65,8011	29	38	0,631579	0,0032896	0,833333

ISS_3d

umbral SalientRadius	Threshold21	Threshold32	NonMaxRadius	tiempo	keypoints1	kepoints2	repetibilidad nube_1	varianza	repetibilidad nube_2
0,00932179	1,5	1,5	0,0186436	2,51764	303	283	0,765677	0,000972631	0
0,0652525	1,5	1,5	0,0186436	17,3554	5833	5840	0,784247	3,91E-005	0,921102
0,172453	1,5	1,5	0,0186436	75,8706	4950	4933	0,772727	6,63E-005	0,949156

SUSAN

umbral radius	distance_thres	angular_thresh	intensity_threshold	tiempo	keypoints1	kepoints2	repetibilidad nube_1	varianza	repetibilidad nube_2
0,00852105	0,00852105	1	7	1,40899	888	926	0,583153	0,000698561	0
0,119295	0,00852105	1	7	198,889	44196	44235	0,760054	2,77E-006	0,966196
0,127816	0,00852105	1	7	236,21	46955	47368	0,756038	1,82E-006	0,970397

En la tabla se muestra los mejores valores con los cuales se obtienen los mejores resultados señalados en color verde.

De forma general,los tiempos de respuestas están relacionados con el umbral, sin embargo la repetibilidad puede variar bajo los diferentes umbrales pero guarda mayor relación con el método. Si el objetivo es determinar los puntos de interés en el menor tiempo posible, con las mejores condiciones de repetibilidad, para elegir el método es necesario minimizar la relación entre el tiempo de cómputo y la repetibilidad como se expresa en la ecuación AIII.2:

$$f(t(u),r)=\frac{t(u)}{r} \qquad \text{(AIII.2)}$$

Siendo *t(u)* los tiempos de cómputo en función del umbral *u*, y *r* es la repetibilidad determinada con una imagen estática perturbada por la acción del ruido en el sensor. Como no podemos obtener una ecuación formal de *t(u)*, pues depende del procesador utilizado y en las condiciones. Se hará uso de las tablas donde se estudiaron los diferentes métodos, bajo las mismas condiciones de procesamiento. En la siguiente tabla se muestra el resumen de los óptimos propuestos para cada método:

Métodos	Min $\|f(t(u),r)\|$	*(u)*
Agast_2d	0.00678965	49
Harris-Tomasi	3.8277824374	0.00932179
Harris-harris	0.7546756004	0.00932179
Haris-Lowe	0.757704733	0.00932179
Harris-Noble	0.74906604857716	0.00932179
ISS_3d	3.2881227985	0.00932179
SUSAN	2.41615836667	0.00852105

Los datos obtenidos en la tabla anterior demuestra que el cálculo de keypoint, por métodos de escala fija, son mucho más lentos que en escala variable. La sencillez de los métodos de escala variable, junto con la necesidad de aportar menor número de datos y menos complejos hace que el proceso de cálculo sea muchos más sencillos. En algunas bibliografías se indican que son más invariantes, algo que es complicado demostrar con la repetibilidad. Se pueden lograr valores similares de repetibilidad, pero esto requiere un tiempo demasiado largo de cómputo.

El mejor método para la obtención de puntos de interés es el de escala fija AGAST_2d. El tiempo de cómputo es escaso y el parche aplicado es suficiente para no estar afectado por el ruido o el efecto de escalonamiento de las imágenes de kinect (que al ser un proceso en 2D no afectaría). Para el resto de métodos, los tiempos que son admisibles tienen radio de umbral demasiado pequeño, en el cual influencia demasiado el efecto de escalonamiento que deja de ser influyente para umbrales 20 veces superior a la resolución media de la imagen 3D.

4 Módulo de Features

En esta sección se describen las clases que contiene PCL para extraer la información de los puntos. Se obtiene, mediante algoritmos, que entregarán descriptores de los puntos de interés. Sin embargo no se realiza comparaciones métricas, como se muestra en la tabla de descriptores las salidas son diferentes entre distintos métodos, solo se pueden considerar los tiempos de cómputo, para establecer criterios de rapidez.

El tipo de descriptor se diferencia principalmente por el enfoque que tienen para obtener la salida de datos. Estos pueden hacer uso de los vecinos más cercanos, siendo estos descriptores locales. También se pueden conseguir descriptores que tienen en cuenta la forma de objetos, denominados descriptores globales. En la siguiente tabla se resumen los tipos de descriptores que son propuestos desde PCL:

Nombre	**Acrónimo**	**Tipo***	**Entrada**	**tamaño****	**Nota**
Point Feature Histogram	PFH	Local (punto)	Puntos + Normales + Método de búsqueda + Radio	16 125 en PCL	-Considera las relaciones angulares entre los pares de puntos, y también tiene en cuenta los vecinos '. - La implementación PCL no utiliza la distancia como una característica (no es discriminatorio).
Fast Point Feature Histogram	FPFH	Local (punto)	Puntos + Normales + Método de búsqueda + Radio	125 33 en PCL	- PFH simplificado sacando las relaciones entre vecinos (simplificado Punto Función histograma). - Para tener en cuenta esto, el SPFH de los vecinos de un punto se fusionó con su propio.
Radius-Based Surface Descriptor	RSD	Local (punto)	Puntos + Normales + Método de búsqueda + Radio+ Radio máximo	289 en PCL	- Estimaciones relaciones radiales entre los puntos (que supone cada par se encuentra en una superficie de la esfera). *No esta implementada por falta de pruebas que validen los datos,
3D Shape Context	3DSC	Local (punto)	Puntos + Normales + Método de búsqueda + Radio+ Radio mínimo + Radio de densidad de puntos	1980 en PCL (varios descriptores)	- Extensión de descriptor 2D. Utiliza una rejilla esférica dividida en sectores a lo largo de cada dimensión. - Se define sin dirección principal, creando varios descriptores para hacer frente a las rotaciones.
Unique Shape Context	USC	Local (punto)	Puntos + Radio + Radio mínimo + Radio de densidad de puntos+ Radio local	1960 en PCL	- Extensión de 3DSC proporcionando una orientación única para cada descriptor para invariancia rotación, - No es necesario el uso de múltiples descriptores.
Signatures of Histograms of Orientations	SHOT	Local (punto)	Puntos + Normales + Radio	Normales 352 en PCL	- Representa rasgos topológicos, tiene puntos dentro de una estructura de soporte esférica. -Invariante a la rotación y traslación, robusto al ruido y el desorden.
Spin Image	SI	Local (punto)	Puntos + Normales + Radio + resolución de imagen	153*** en PCL (ningún tipo de encargo)	- Utiliza una alineación normal de estructura de soporte cilíndrica, dividida en volúmenes. - El descriptor más antiguo en PCL.
Rotation-Invariant Feature Transform	RIFT	Local (punto)	Puntos + Color + Radio +Gradiente de intensidad	32*** (ningún tipo de encargo)	- extensión de 2D SIFT, usa la información de textura (color). Utiliza una máscara circular. - Vulnerable a los giros.

Nombre	Acrónimo	Tipo*	Entrada	tamaño**	Nota
Normal Aligned Radial Feature	NARF	Local (punto)	Imagen de rango + Keypoints + tamaño de soporte	36	- Extensión de 2D SIFT. Usualmente los keipoint están cerca de las esquinas.. NARF codifica la curvatura alrededor. - Trabaja con imágenes de rango, no con nubes. Necesita extraer las fronteras para encontrar los keypoints.
Rotational Projection Statistics	RoPS	Local (punto)	Puntos + Triángulos + Radio soporte+ Número de rotaciones y cubos	ningún tipo de encargo)	-Requiere un paso previo de triangulación, ya que funciona con una malla de triángulos, en lugar de la nube. - Recopila información estadística sobre la distribución alrededor de los puntos, proyectada sobre los ejes.
Viewpoint Feature Histogram	VFH	Global (objeto)	Puntos + Normales + Método de búsqueda	263 308 en PCL	- Estudia el ángulo entre la normal del punto y el centroide de la nube - Deriva de FPFH, utiliza una simplificación de SPFH además de un componente de visión-dependiente.
Clustered Viewpoint Feature Histogram	CVFH	Global (objeto)	Puntos + Normales + Método de búsqueda + Umbral de ángulo+ Umbral de curvatura	308 en PCL (varios descriptores, usa tipo VFH)	- Divide el objeto en N regiones lisas disjuntas y calcula el VFH de cada uno. - Robusto a la oclusión, siempre y cuando la región es visible, opcionalmente puede ser dependiente de la escala.
Oriented, Unique and Repeatable Clustered Viewpoint Feature Histogram	OUR-CVFH	Global (objeto)	Puntos + Normales + Método de búsqueda + Umbral de ángulo+ Umbral de curvatura	308 en PCL (varios descriptores, usa tipo VFH)	- mejora el CVFH con el cálculo de un marco de referencia único para hacerlo más robusto. - Además mejora la región de segmentación.
Ensemble of Shape Functions	ESF	Global (objeto)	Puntos	640 en PCL	- Describe de la nube (distancias, ángulos, áreas). Robusto a errores como los valores extremos, el ruido, agujeros, ...
Global Fast Point Feature Histogram	GFPFH	Global (objeto)	Puntos + Etiquetas + Número de clases+ Tamaño de las hojas	16 en PCL	- Versión global de FPFH. - Utiliza una etapa de clasificación en o la superficie anterior, que etiqueta puntos como pertenecientes a una de las clases.
Global Radius-Based Surface Descriptor	GRSD	Global (objeto)	Puntos + Normales + Método de búsqueda	21 en PCL	- Versión global de RSD. Al igual que GFPFH, utiliza la categorización de superficie y una rejilla.

* Las características locales se estiman para cualquier punto (normalmente un conjunto submuestreados de keypoints). Las características globales se estiman para un grupo de puntos que se cree que es un objeto,lo que requiere un paso previo de preprocesamiento(segmentación).
** Se representa las dimensiones de la firma de los descriptores (tamaño de histograma). Si está disponible, también se incluye el tamaño utilizado en la publicación original. Algunos descriptores no tienen una medida para PointType en PCL.
*** *El tamaño de este descriptor varia en función de los parámetros elegidos y el valor dado es valido cuando se utiliza ciertos valores predeterminados.*

Las estructuras de los datos entregados por cada descriptor están formadas por vectores de longitud limitada. Esta será función del algoritmo utilizados. En el caso de PCL se tienen las siguientes estructuras de datos de descriptores locales:

```
struct FPFHSignature33 {
        float histogram[33];
        friend std::ostream& operator << (std::ostream& os, const FPFHSignature33& p);
            };
struct PFHSignature125 {
        float histogram[125];
        friend std::ostream& operator << (std::ostream& os, const PFHSignature125& p);
            };
```

```
struct ShapeContext1980 {
                float descriptor[1980];
                float rf[9];
                friend std::ostream& operator << (std::ostream& os, const ShapeContext1980& p);
                        };
struct SHOT352 {
                float descriptor[352];
                float rf[9];
                friend std::ostream& operator << (std::ostream& os, const SHOT352& p);
                  };

template < int N>
struct Histogram{
                float histogram[N];
                   };
struct Narf36 {
                float x, y, z, roll, pitch, yaw;
                float descriptor[36];
                friend std::ostream& operator << (std::ostream& os, const Narf36& p);
};
```

El módulo en el que se implementan los distintos algoritmos para la obtención de descriptores, tienen en común, que están diseñados para la estimación de unos poco puntos de la nube mediante la función setInputCloud( ), todos son introducidos mediante la función setSearchSurface(). En el caso de que se estudien parámetros geométrico, lo más habitual es hacerlo con la características implícitas de la nube, que son los vectores normales del punto y su curvatura, introducidos mediante la función setInputNormals(). La máscara o umbral es el parámetro que se necesita para definir el alcance de influencia, sobre el que trabajará el algoritmo, y será el radio de la esfera con origen en el punto de estudio, donde se extraerán para obtener la información local del punto mediante la función setRadiusSearch(). Otros parámetros son propios de cada clase u opción.

El parámetro de la máscara es muy importante seleccionarlo de forma que los errores de tolerancia del sensor no influyen sobre el resultado. El efecto sobre las normales puede quedar atenuado, los filtros que se aplican no son capaces de corregir estos efectos. Tampoco es deseable que se introduzcan máscaras muy grandes, sino la información dejará de ser local, y se ralentiza el proceso de cálculo

Para la elección de un método de obtención de descriptores, realizando la búsquedas de keypoint o puntos de interés con una “supresión de puntos” igual a la mascará utilizada, en las normales, se evitan los defectos de la imagen que influyan en la obtención de los descriptores. Por otro lado, aplicaremos máscaras con radio, al menos, superior al utilizado en las normales. No se discute cuál es el mejor método por la forma de obtener los descriptores, o el tamaño de estos, pero sería conveniente obtenerlos en el menor tiempo posible. En la siguiente tabla se establece la relación de tiempos entre los diferentes descriptores y los métodos de obtención de los keypoint, los

tiempos estarán en función del número de keypoints resultante.

	PFH	FPFH	3DSC	USC	SHOT	SI
ISS_3d	0.608921	2.44374	0.889276	0.198157	0.15698	0.024829
SUSAN	0.02427	0.035738	0.034508	0.034115	0.02785	1e-05
HARRIS	0.010251	0.058822	0.008581	0.027686	0.025884	9e-06
NOBLE	0.010578	0.062565	0.008678	0.025373	0.026713	8e-06
LOWE	0.010451	0.046911	0.008514	0.025422	0.024365	7e-06
TOMASI	0.062925	0.193775	0.047732	0.034624	0.029487	8e-06
Agast-2D	0.187069	0.40828	0.188173	0.058052	0.045085	9e-06

Tabla de tiempos de cómputo de descriptores para diferentes subconjuntos de keypoints.

5 Módulo de registro

El registro engloba un conjunto de conceptos que se aplican por separados, pero persigue un mismo objetivo, es el solape de las nubes de puntos. El módulo de registro lo realiza por medio de comparaciones entre puntos de una nube con otra que pertenece a parte de una misma realidad. Este tiene diferentes clases que podemos agrupar en diferentes tipos según sus funciones, de los cuales destacamos los siguientes:

1. Correspondencias entre puntos.

2. Rechazo de correspondencias.

3. Obtención de Transformación.

Esta clasificación describe cada uno de los procesos de la etapa final del registro. No son necesarios realizar los dos primeros pasos para finalizar con éxito, pero se estudiarán pues son los que se recomienda en la Librería PCL.

5.1 Correspondencias entre puntos.

La correspondencia es un proceso que consiste en emparejar puntos de diferentes nubes, bajo un criterio, en el que se supone que ambos puntos de diferentes nubes pertenecen a un mismo punto de la realidad. En esta librería solo existe una clase que realiza esta tarea y es la siguiente:

pcl::registration::CorrespondenceEstimation< PointSource, PointTarget, Scalar >

Lo importante de esta clase es la información que se aporta para que realice las comparaciones, se pueden aportar los datos de profundidad de los keypoints. Aunque recomienda aportar la información de los descriptores. Esta clase soporta los siguientes tipos de datos:

1. PointXYZ~
2. FPFHSignature33
3. PFHSignature125
4. ShapeContext1980
5. SHOT352

Los descriptores presentados contienen la información local del punto, la clase empareja aquellos puntos que tienen descriptores con valores similares. Como se observa hay descriptores que contienen más datos que otros, debe traducirse en información. Sin embargo, cuando se aplica esta clase con los descriptores, los emparejamientos son muy dispares, se puede observar en la siguiente figura. En esta se representan los keypoints de dos nube de puntos, tienen los puntos atenuados y sus diferentes keypoints resaltados en rojo para una nube y para la otra en verde. Estos tienen las correspondencias representadas mediante líneas azules para aquellos casos que suponemos que no son válidas y amarillo para las que si.

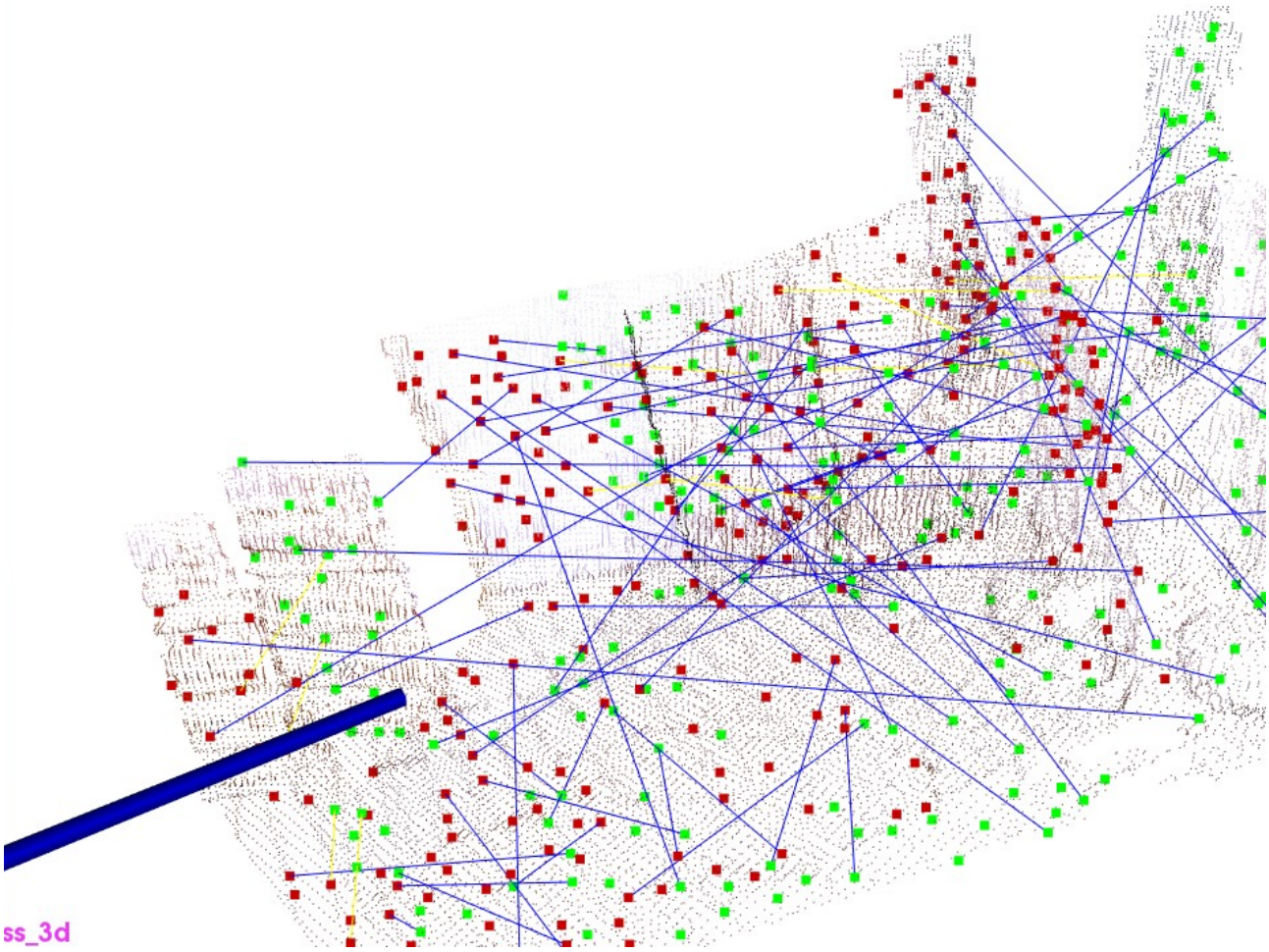


Figura AIII.13: nubes de puntos con líneas de correspondencias entre keypoints

Que se obtengan tantas correspondencias no válidas puede ser debido a que se tomen solo parámetros locales para realizarlas, ya que se supone que pueden tener orígenes muy distantes.

La estructura de datos de correspondencias no toma el punto sino que toma el índice de la nube, de los puntos de interés. Por tanto, se genera un vector con pares de índices que son las correspondencias y opcionalmente se puede tener la distancias del emparejamiento, como la ponderación del mismo en el conjunto de correspondencias. La estructura sería de la siguiente forma:

```
struct Correspondence {
    int index_query;
    int index_match;
    union
    {
        float distance;
        float weight;
    };
    inline Correspondence () : index_query (0), index_match (-1),
        distance (std::numeric_limits<float>::max ())
    {}
    inline Correspondence ( int _index_query, int _index_match, float _distance) :
        index_query (_index_query), index_match (_index_match), distance (_distance)
    {}
    virtual ~Correspondence () {}
    EIGEN_MAKE_ALIGNED_OPERATOR_NEW
};
```

5.2 Rechazo de Correspondencias.

La solución al problema de correspondencias inválidas se realiza mediante un filtrado, teniendo en cuenta otros parámetros diferentes a los aplicados en el paso de correspondencias. Obviamente no se puede comparar con otro tipo de descriptor, la información que aporta es muy similar. Para realizar los rechazos de correspondencias la librería PCL dispone de las siguientes clases:

pcl::registration::CorrespondenceRejectorDistance

En esta clase se realiza el rechazo de correspondencias mediante la umbralización de las distancias entre las correspondencias. Este método es válido cuando los orígenes de ambas nubes son cercanos y sin grandes rotaciones, quiere decir que un punto de la realidad solo ha sido desplazado una distancia corta.

pcl::registration::CorrespondenceRejectorMedianDistance

Al igual que en la clase anterior, se tiene en cuenta la distancia entre correspondencias. Pero se estima la distancia media de todas las correspondencias y se aplica un umbral sobre la mediana. Esta no es adecuada si se aplican registros sobre nubes donde se han producido grandes giros, aunque tiene su utilidad en pasos intermedios de alineación.

pcl::registration::CorrespondenceRejectorOneToOne

En algunos casos puede ocurrir que un mismo punto tenga dos correspondencias con los puntos de otra nube, repitiendo un mismo índice, por lo que es evidente que al menos una de las correspondencias no es válida para el registros. La clase mantiene la correspondencia de menor distancia eliminando el resto que comparten índice.

pcl::registration::CorrespondenceRejectorFeatures

Se puede rechazar las correspondencias mediante descriptores, en esta clase se toman los puntos de y se comparan ambos descriptores definiendo un umbral de tolerancias en el espacio de los descriptores.

pcl::registration::CorrespondenceRejectorSampleConsensus< PointT >

El método de rechazo consiste en eliminar los valores atípicos de un conjunto de muestras aleatorias. Este tipo no se recorre todos los datos, sino que se toma parte de estos, se analizan y realiza la eliminación si procede. Si el número de iteraciones es muy pequeño pueden no eliminarse todos los datos atípicos.

pcl::registration::CorrespondenceRejectorSurfaceNormal

En muchos casos llegados al filtrado de correspondencias, se han calculado previamente las normales de las nubes, con esta clase se aprovecha esta información para comparar el ángulo entre ellas.